2011

Implementation and Qualitative/ Quantitative Comparison and Evaluation of Range Flow and Scene Flow

Saima Sarwary

Implementation and Qualitative/ Quantitative Comparison and Evaluation of Range Flow and Scene Flow

(Spine title: Evaluation of Scene and Range Flow)

(Thesis format: Monograph)

by

Saima Sarwary

Graduate Program in Computer Science

Submitted in partial fulfillment
of the requirements for the degree of
Master of Science

School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario
August, 2011

THE UNIVERSITY OF WESTERN ONTARIO
School of Graduate and Postdoctoral Studies

CERTIFICATE OF EXAMINATION

Supervisor	Examiners
Dr. John Barron	Dr. Steven Beauchemin
	Dr. Sylvia Osborn
	Dr. Andre Boivin

The thesis by
Saima Sarwary

entitled

IMPLEMENTATION AND QUALITATIVE/ QUANTITATIVE COMPARISON AND EVALUATION OF RANGE FLOW AND SCENE FLOW

is accepted in partial fulfillment of the
requirements for the degree of
Master of Science

Date

Chair of Examining Board

Abstract

This thesis investigates the use of 3D scene flow and 3D range flow as a means of computing 3D observer (sensor or camera) motion. We implemented and evaluated the scene flow algorithm presented in Stereoscopic Scene Flow Computation for 3D Motion Understanding by Wedel et al. 2010. We modified (performed pyramidal image processing with warping) and re-implemented the range flow algorithms presented in Quantitative Regularized Range Flow by Spies et al. 2000. Both algorithms are 2-frame algorithms using a pyramid. The results for these scene and range flow algorithms were quantitatively and qualitatively compared on synthetic and real car driving stereo sequences.

Keywords: range flow; scene flow; least squares; regularization; disparity; depth maps; warping; pyramid;

Acknowledgments

This research would not have been possible without the support of many people.

I would like to express my heartfelt gratitude and respect to my thesis supervisor, Dr. John Barron, for his tremendous help, encouragement, guidance, patience, and support from the initial phase of the project until the final phase, which lead to a successful completion of my thesis. Without a doubt, Dr. Barron is one of the best supervisors anyone could ever hope for. He gave countless hours of help towards solving my problems, answering my questions and helping me debug my code. Over the course of my research, I have learned a great deal about computer vision, MatLab, and Latex from him and also developed a thorough understanding of the topic.

I am also thankful to my course professors, John Barron, Yuri Boykov and Steven Beauchemin, who taught me Computer Vision and Image Processing. Special thanks to Mathworks.com and Andreas Wedel and Reinhard Klette of the University of Auckland for answering our questions by email.

I am also deeply grateful to my mother for her constant love, financial, moral, and mental support and encouragement to follow my interests and dreams and excel in life. Special thanks to my brother, aunt and uncle and my friends for their love, help, and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

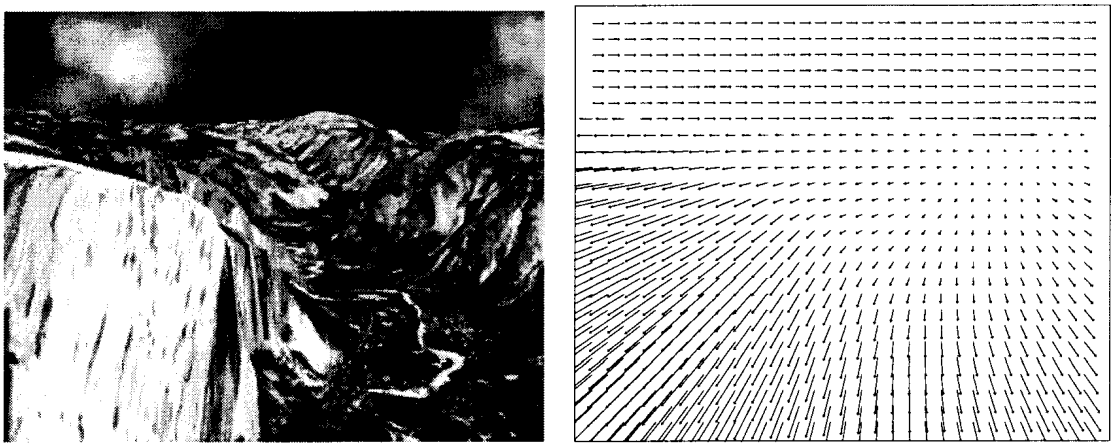
This thesis is concerned with the computation of 3D range flow and 3D scene flow. Both range and scene flow compute the same thing (3D sensor motion) relative to 3D environmental points). Before we can explain these terms more fully we need to define 2D and 3D optical flow first. This thesis presents the results of one algorithm from each of the 2 approaches and compares them on the same synthetic and real data. Synthetic data allows a quantitative analysis while real data allows only a qualitative analysis. We first define 2D/3D optical flow, 3D scene flow and 3D range flow in this chapter. Then we outline the thesis topics and give a summary of the contributions of the thesis. Finally, an outline of what to expect in the coming chapters is provided.

1.1 2D Optical Flow, 3D Optical Flow, Range Flow, and Scene Flow

This section explains the meaning of 2D/3D optical flow, 3D range flow and 3D scene flow.

Optical flow is a significant research area in Computer Vision and Image Processing. Optical flow, also known as image velocity, is the apparent visual motion that

you experience as you move through the world. One example of optical flow is while sitting in a moving car and looking out the window, you see that trees, the ground, buildings, etc., appear to move backwards. The speed of this motion depends on the distance of these objects from you. Far away objects move slower and close objects move faster. Individual motions are called optical flow or image velocities. The entire motion field is called the optical flow field. Figure 1.1 shows an example of a synthetic image (the 8^{th} image of the Yosemite fly through sequence) and its correct 2D optical flow field. We see vectors at each pixel, telling us where that pixel will move to in the next image.



(a) (b)

Figure 1.1: (a) The middle frame from the **Yosemite Fly-Through** *sequence and (b) its correct flow field.*

Optical flow methods try to calculate the motion between two image frames which are taken at times t and $t + \delta t$, meaning that optical flow specifies how much each pixel moves between two adjacent image frames. There are many methods to compute optical flow and some of these methods are explained in detail in Chapter 2. Optical flow is important for motion estimation, video compression, object detection and tracking, image dominant plane extraction, movement detection, robot navigation and visual odometry. Some useful applications of optical flow include traffic analysis

and vehicle tracking.

However, optical flow provides only projected 2D information so this has some drawbacks. For example, there are ambiguities when dynamic (or moving) 3D objects are explained using 2D optical flow. 3D optical flow can be calculated on 3D images (3D volumes of data). 3D optical flow specifies how much each voxel moves between adjacent volumes in the dataset. Barron [2] showed how 3D optical flow could be extended from 2D optical flow and this will be explained further in Chapter 2.

There are two types of 3D optical flow called range flow and scene flow. They are the same 3D concept, but range flow is computed from a depth map and its spatio-temporal derivatives while scene flow is computed from a disparity map (and its gradient map) as well as the 2D optical flow of the left and right images in a stereo image sequence. Both scene flow and range flow can be described as the 3D optical flow on visible environmental object surfaces. Scene flow is both depth (disparity) and intensity based while range flow (in its purest form) involves depth based only (but we have added intensity data to our calculations). If a scene is rigid and stationary and only the sensor (camera) is moving, then correct 3D scene flow and 3D range flow will just be a constant 3D vector at each pixel. Of course, computed 3D scene flow and 3D range flow may have different 3D vectors for each pixel due to various types of error (although generally they should be similar).

This thesis describes the implementation and evaluation of the scene flow algorithm presented in papers entitled "Efficient Dense Scene Flow from Sparse or Dense Stereo Data" and "Stereoscopic Scene Flow Computation for 3D Motion Understanding" by Andreas Wedel, Clemens Rabe, Tobi Vaudrey, Thomas Brox, Uwe Franke and Daniel Cremers in 2008 and 2010 [29, 28] and the range flow algorithms presented in papers entitled "Differential Range Flow Estimation", "Dense Range Flow from Depth and Intensity Data",, "Quantitative Regularized Range Flow", "Regularized Range Flow" and "Range Flow Estimation" by Hagen Spies, John Barron and Bernd Jahne from 1999 to 2002 [23, 4, 22, 24]. Our results for scene and range flow were

quantitatively and qualitatively compared on a number of different synthetic and real stereo image sequences (of cars driving on road environments). The 2010 Wedel et al. IJCV paper and the Spies et al. 2000 ICPR paper are the main papers.

1.2 Thesis Contributions

The contributions of the thesis are summarized below.

1. We implemented and evaluated the scene flow algorithm presented in Stereoscopic Scene Flow Computation for 3D Motion Understanding by Wedel et al. 2010 [28] in MATLAB. Since MATLAB is optimized for matrix operations, we vectorized our programs as much as possible to gain higher efficiency. We also use the same differentiation method as proposed by Brox et al. 2004 [6] (namely simple pixel differences for temporal derivatives and 4-point central differences for spatial derivatives). We tested this algorithm on synthetic and real car sequences and on synthetic sinusoid data.

2. We derived new range flow equations for perspective projection based on the work by Spies and Barron [24], since their old equations were for orthographic projection only. We modified and re-implemented the least squares and regularized range flow algorithms presented by Spies et al. [24] in the same pyramidal data structure as used by Wedel et al. for his scene flow algorithm.

3. We displayed the correct 3D scene/range flow field using vector fields instead of colour images. When we do this, we see that the colour images often "hid" problems in the flow fields that are very visible in the vector fields.

4. We performed a quantitative analysis using two synthetic stereo car image sequences and one (out of seven) real Daimler car driving sequences our scene and range flow implementations.

5. We showed the need to robustify both the least squares and regularization solution frameworks. As future work, we plan on using Brox et al.'s [6] robust estimation framework (as was used in Wedel et al.'s algorithm presented here). There are simply too many outliers for the range flow results to be considered good.

6. We showed some problems with respect to the correct car flow data (the presence of outlier data) requiring us to "clean" the data. We also observed some problems for Wedel et al.'s scene flow algorithm with respect to warping (for example, we found 2 typos in the sor-step (algorithm 5) which Wedel has confirmed). We don't know if our other problems are a result of our not fully understanding their algorithm or a lack of detail in their journal paper (especially about warping). We communicated with Wedel by email about our problems but were unable to resolve them. Wedel et al.'s code is not available.

7. We verified the mathematical equivalency of range flow and scene flow in Chapter 3. We also verified that u, v, p can be inversely solved from U, V, W in that chapter. U, V, W is the 3D range or 3D scene flow velocities while (u, v) is the 2D optical flow and p is the change in disparity. Note that if we have computed U, V, W from range data, then u, v, p are for the equivalent scene (imaginary) data.

8. We introduced a primitive warping technique for range flow in the Z pyramid by removing the incremental change dW from the depth maps at each level before projecting the Z values to the next level. This slows down the motion at lower levels and reduces the possibility of aliasing.

1.3 Outline of Thesis

In this chapter, we defined the basic concept of optical/scene/range flow, provided an outline of the thesis, and listed the thesis contributions. The rest of the thesis is organized as follows. Chapter 2 provides a literature survey of range and scene flow algorithms. To keep this survey manageable we only look at optical flow algorithms relevant to range and scene flow as referenced by Spies et al. and Wedel et al. In Chapter 3, we discuss the two implemented algorithms by Wedel et al. [28] and Barron and Spies [4] in detail. Chapter 4 explains the methods/tools we used to implement and evaluate the two algorithms. In Chapter 5, we present and discuss our experimental results. We end the thesis with conclusions and future work in Chapter 6.

Chapter 2

Literature Survey

This chapter provides background information on relevant 2D optical flow and 3D optical flow algorithms before we review 3D range flow and 3D scene flow algorithms relating to this thesis. We summarize many of the papers relevant to Wedel et al.'s work as outlined their 2010 IJCV paper [28] and to Spies et al.'s work as outlined in their 2002 CVIU paper [24]. We present Wedel et al's and Spies et al.'s algorithms in detail in Chapter 3.

2.1 Overview

To understand the basics of optical flow, we review relevant papers on optical flow cited in Wedel et al. 2010 [28], such as Horn and Schunck 1981 [13] and Brox et al. 2004 [6]. Also, some concepts in Longuet-Higgins and Prazdny 1980 [17], and an implementation and evaluation of Brox et al.'s 2D optical flow algorithm work by Faisal and Barron 2007 [8] are presented below.

Scene flow can be categorized into two kinds of algorithms:

1. Motion and disparity estimation methods are performed simultaneously as in

Huguet and Devernay 2007 [14], Rabe et al. 2007 [20], Isard and MacCormick 2006 [15], Zhang et al. 2001 [30], and Patras et al. 1997 [19] and

2. Position and velocity estimation steps are decoupled in the algorithm as in Wedel et al. 2008 [29] and Wedel et al. 2010 [28].

Range flow algorithms can be categorized in two kinds of algorithms:

1. Motion estimation of locally rigid objects moving in an environment observed by a stationary sensor such as in Spies et al. 2002 [24], Spies et al. 1999 [23], Spies et al. 2000 [10], and the regularized range flow algorithm in Spies et al. 2000 [22] and Barron and Spies 2000 [4] and

2. Motion estimation of globally rigid objects as in Harville et al. 1999 [12].

The relevant aperture problems associated with 2D/3D optical flow, 3D scene flow and 3D range flow are also presented below.

2.2 2D Optical Flow

2D optical flow methods calculate the motion or displacement of pixels between two image frames acquired at times t and $t + \delta t$[1]. It is assumed that the scene lighting is Lambertian (no specularities), that intensity variations between adjacent images are due to the motion of the camera with respect to the scene and not due to other artifacts such as changing illumination or deforming objects, that local motion can be well approximated by pure translation and that image intensity derivatives are due to motion only. In as much as these assumptions are satisfied, optical flow is a good approximation to the local 2D motion motion in image sequences. Basic concepts needed to explain optical flow, such as how to project a 3D world point onto a 2D

[1]We will ignore differential optical flow using more than 2 frames.

image plane and the derivation of the image velocity (optical flow) equations in terms of 3D sensor velocity and environmental depth are explained in detail in Longuet-Higgins et al. 1980 [17]. In this thesis, we are only concerned about differential based optical flow. That is, we use intensity derivatives in the computation of optical flow.

A pioneer method to calculate optical flow is the 1981 algorithm presented by Horn and Schunck [13]. A basic assumption in this algorithm is that if a pixel moves from one location to another, the grayvalues or intensities in the neighbourhood of that pixel remains constant. In other words, even if an object changes position during the short time interval from t_1 to t_2, the reflectivity and illumination of the object will remain the same. Mathematically, this can be expressed as

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t), \tag{2.1}$$

where $I(x, y, t)$ is the intensity of the image at position (x, y) and time t, while $\delta x, \delta y$ is the change in position of that pixel over time δt. This leads to the 2D motion constraint equation, whose derivation is shown in the next section. We also discuss the aperture problem.

2.2.1 2D Motion Constraint Equation

We derive the 2D motion constraint equation in this section. Assume $I(x, y, t)$ moves by δx and δy in time δt to $I(x + \delta x, y + \delta y, t + \delta t)$ as shown in Figure 2.1 below.

Because $I(x, y, t)$ and $I(x + \delta x, y + \delta y, t + \delta t)$ are the images of the same point, we can write:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t), \tag{2.2}$$

. If we perform a 1^{st} order Taylor series expansion about $I(x, y, t)$, we get:

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + H.O.T. \tag{2.3}$$

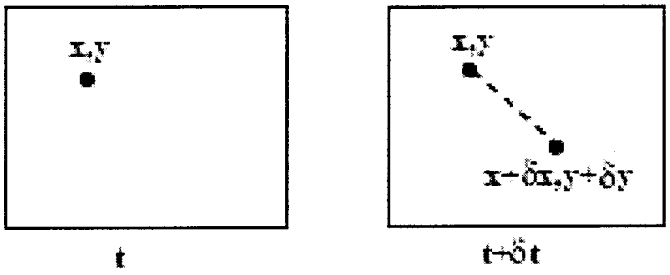


Figure 2.1: A pixel's displacement from (x, y) to $(x + \delta x, y + \delta y)$ from time t to time $t + 1$.

We can ignore the higher order terms, $H.O.T$, as we are dealing with a small neighborhood, and because $I(x, y, t) = I(x+\delta x, y+\delta y, t+\delta t)$ we substitute Equation (2.2) into Equation (2.3) and obtain the motion constraint equation:

$$\begin{aligned} \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t &= 0, \\ \frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t}\underbrace{\frac{\delta t}{\delta t}}_{=1} &= 0, \\ I_x u + I_y v + I_t &= 0 \end{aligned}$$

$$\nabla I \cdot \vec{v} + I_t = 0. \tag{2.4}$$

Here $u = \frac{\delta x}{\delta t}$ and $v = \frac{\delta y}{\delta t}$ are the x and y components of image velocity, $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$ and $I_t = \frac{\partial I}{\partial t}$ are image intensity derivatives at $I(x, y, t)$ and $\vec{v} = (u, v)$. The motion constraint equation is also sometimes referred to as the optical flow constraint equation.

2.2.2 2D Aperture Problem

One important problem to be overcome when computing 2D optical flow is the 2D aperture problem (points for which optical flow cannot be determined) which is ex-

plained in Horn and Schunck 1981 [13]. The aperture problem can be explained in terms of the motion constraint equation line. Intuitively, the aperture problem means that only the component of the flow perpendicular to the local intensity structure (in the direction of the intensity gradient) can be computed while the tangential component of the flow cannot be computed. In order to compute full velocity or full optical flow additional constraints must be incorporated into the solution.

That is, $\nabla I \cdot \vec{v} + I_t = 0$ is 1 equation with 2 unknowns (the line described by the motion constraint equation), as shown below in Figure 2.2 and the correct velocity is some unknown point on this line. The velocity with the smallest magnitude that is on this line is called the normal velocity $\vec{v}_n = (u_{nx}, v_{ny})$. Note that a line from the origin to that point is perpendicular or normal to the motion constraint line.

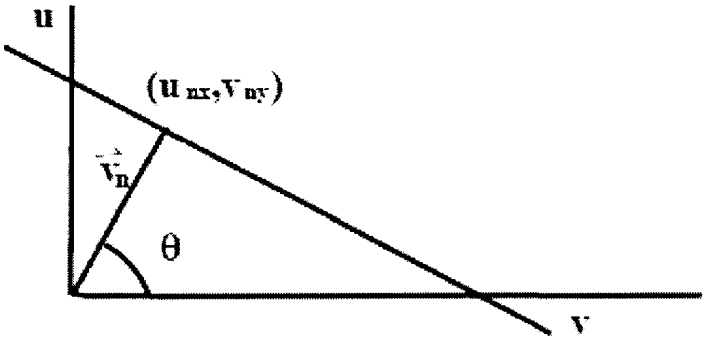


Figure 2.2: The motion constraint line. Normal velocity $\vec{v}_n = (u_{nx}, v_{ny})$ is the velocity with the smallest magnitude that is on this line.

The relationship between the motion constraint equation and normal velocity can be described as follows. Consider a straight contour moving up/down and to the right with true full velocity $\vec{v}$. Since it is viewed through an aperture we can only see the motion perpendicular to the contour. Since the normal velocity is the smallest of all potential velocities it is the point on the motion constraint line closest to the origin. A line from that point to the origin is normal to the motion constraint line.

We can compute v_n (the magnitude of $\vec{v}_n$) and $\hat{n}$ (its direction) and hence, the normal velocity, $\vec{v}_n = v_n\hat{n}$, very simply using the motion constraint equation $\nabla I \cdot \vec{v} = -I_t$ and the fact that $\vec{v} \cdot \hat{n} = v_n$ as:

$$\begin{aligned} \nabla I \cdot \vec{v} &= -I_t \\ \frac{\nabla I}{||\nabla I||_2} \cdot \vec{v} &= -\frac{I_t}{||\nabla I||_2} \\ \hat{n} \cdot \vec{v} &= v_n, \end{aligned}$$

where

$$\Rightarrow \hat{n} = \frac{\nabla I}{||\nabla I||_2} \quad \text{and} \quad v_n = \frac{-I_t}{||\nabla I||_2}. \tag{2.5}$$

Methods to overcome the aperture problem and solve for full optical flow require that additional constraints must be brought to bare on the problem. For example, Lucas and Kanade 1981 [18] assume than velocity is constant in local neighbourhoods (but normal velocity is not). In that case, 2 or more different normal velocities yield the true full velocity (in the least squares sense). On the other hand, Horn and Schunck 1981 [13] assumed that velocity varies smoothly everywhere and regularized or minimized a smoothness term across the image. Detailed descriptions of these two algorithms are given below. Other, state-of-the-art methods, include Brox et al. 2004 [6], where larger displacements were dealt with using image warping in a pyramidal data structure and a nonlinearized robust regularization model. This computational framework is used by Wedel et al. and presented below.

2.3 Horn and Schunck 1981 [13]

Horn and Schunck's [13] algorithm estimates optical flow between two images. An energy function assumes constant pixel intensities and a smooth flow field and this function is minimized using local variational optimization. In other words, the al-

gorithm assumes that even if a pixel moves from one location to another, the gray values of the pixel remain constant. This assumption leads to the motion constraint equation or the optical flow constraint equation: $I_x u + I_y v + I_t = 0$. This means that if a local image patch is translating with velocity (u, v), then this local velocity is constrained by the motion constraint equation.

To deal with the aperture problem, Horn and Schunck suggest a variational method which uses an additional smoothness term to yield a 100% dense optical flow field. This additional smoothness constraint minimizes the square of the magnitude of the gradient of the optical flow velocity. The smoothness term is:

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2. \tag{2.6}$$

Horn and Schunck proposed a scheme to estimate the derivative of brightness measurements from two images only (a weighted difference calculation). However the flow field turns out much better if Simoncelli derivatives [21] are used, requiring an image sequence containing 7 images [3]. The functional:

$$\int\int I_x u + I_y v + I_t(x, y) + \lambda^2 \left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2\right] dxdy \tag{2.7}$$

is minimized using the Gauss Seidel method to iteratively minimize the Euler-Lagrange equation to produce a globally smooth optical flow field. The parameter λ is a regularization constant (sometimes called the Lagrange multiplier) and larger values of λ lead to a smoother optical flow, λ^2 is a weighting factor (squaring λ guarantees the term is always positive) between the brightness constraint and smoothness constraint; larger values mean more smoothing.

The minimization of this function will yield the optimal optical flow velocity (u, v). Using Gauss Seidel and the Euler-Lagrange equations, a solution for u and v can be found iteratively. It would be very costly and probably impossible (due to roundoff

error) to solve these linear equations simultaneously by one of the standard methods, such as Gauss-Jordan elimination. Horn and Schunck compute a new set of velocity estimates (u^{n+1}, v^{n+1}) from the estimated derivatives and the average of the previous velocity estimates (u^n, v^n) as:

$$u^{n+1} = \bar{u}^n - I_x[I_x\bar{u}^n + I_y\bar{v}^n + I_t]/(\alpha^2 + I_x^2 + I_y^2) \quad (2.8)$$

$$v^{n+1} = \bar{v}^n - I_x[I_x\bar{u}^n + I_y\bar{v}^n + I_t]/(\alpha^2 + I_x^2 + I_y^2). \quad (2.9)$$

The averages of u and v are usually initialized to 0 for each pixel. Eventually, the difference in the optical flow between two successive iterations will become very small (less than some preset threshold τ), at which point we assume that we reached the optimal solution.

One of the advantages of the Horn Schunck's algorithm include that it yields a 100% dense flow field. But, on the other hand, it often smoothes out discontinuities in the flow field. Horn and Schunck's algorithm cannot deal with large displacements, and this problem was tackled using a hierarchical approach with image warping and non-linearized model equations in Brox et al. 2004 [6] (they were not the first to use a hierarchical approach, see Bergen et al. [5] for an earlier hierarchical approach).

2.4 Brox et al. 2004 [6]

The best results in terms of accuracy (still to this day) were obtained by Brox et al. 2004 [6]. They avoided the linearizaton of the different energy terms in the variational formulation by warping (see item 4 on the next page) the image at time $t+1$ onto the image at time t, and they only linearized the global energy inside the minimization algorithm. To compute optical flow, they proposed an energy functional that combined three assumptions: brightness constancy, gradient constancy and a discontinuity-preserving spatio-temporal smoothness constraint. They included the

following constraints in their model:

1. Grayvalue constancy assumption, which means that the intensity/brightness/gray value of a pixel is not changed by the displacement, which leads to the image constraint equation, $I(x, y, t) = I(x+u, y+v, t+1)$, where $\vec{v} = (u, v)$. This equation is nonlinearized. The linearised version of the grey value constancy assumption using Taylor expansion yields the famous optical flow constraint/motion constraint equation: $I_x u + I_y v + I_t = 0$. This equation is true or the linearizaton is valid only when the image changes linearly along the displacement, which is not held in case of large displacements. This model uses the nonlinearized equation. This function is quite susceptible to slight changes in brightness, which often appear in natural scenes.

2. Gradient constancy assumption, $\nabla I(x, y, t) = \nabla I(x + u, y + v, t + 1)$. Here $\nabla = (\partial_x, \partial_y)^T$ is the spatial gradient. This function requires that the spatial gradient does not vary due to displacement. This criterion is invariant under grayvalue changes.

3. Smoothness assumption, which is based on the Horn and Schunck [13] smoothness assumption with quadratic penalisers. The spatio-temporal gradient which indicates a spatio-temporal smoothness assumption is involved for applications with more than two images.

4. Pyramid coarse to fine warping is used. Warping (using the computed flow to warp the 2nd image towards the 1st image at each level in the pyramid) removed the current flow from the image. Thus as processing continues down the pyramid towards the original image, the flow between the 1st image and the warped 2nd image becomes smaller and smaller.

The grayvalue and gradient assumptions are measured by the energy:

$$E_{Data}(u,v) = \int_{\Omega} \Psi(|I(x+u,y+v,t+1) - I(x,y,t)|^2) \tag{2.10}$$

$$+\gamma(|\nabla I(x+u,y+v,t+1) - \nabla I(x,y,t)|^2)dxdydt. \tag{2.11}$$

Here Ψ does the robust estimation, $\Psi(s^2) = \sqrt{s^2+\epsilon}$ and ϵ= 0.001.

The smoothness term that models the assumption of piecewise smoothness is:

$$E_{smooth}(u,v) = \int_{\Omega} \Psi(|\nabla_3 u|^2 + |\nabla_3 v|^2)dxdydt. \tag{2.12}$$

Here $\nabla_3 = (\partial_x, \partial_y, \partial_t)^T$ is the spatio-temporal gradient. The total energy is:

$$E(u,v) = E_{Data} + \alpha E_{smooth}. \tag{2.13}$$

For better readability, they define the following abbreviations:

$$I_x = \partial_x I(x+w) \tag{2.14}$$

$$I_y = \partial_y I(x+w) \tag{2.15}$$

$$I_z = I(x+w) - I(x) \tag{2.16}$$

$$I_{xx} = \partial_{xx} I(x+w) \tag{2.17}$$

$$I_{xy} = \partial_{xy} I(x+w) \tag{2.18}$$

$$I_{yy} = \partial_{yy} I(x+w) \tag{2.19}$$

$$I_{xz} = \partial_x I(x+w) - \partial_x I(x) \text{ and} \tag{2.20}$$

$$I_{yz} = \partial_y I(x+w) - \partial_y I(x). \tag{2.21}$$

According to the calculus of variations, a minimizer of the equation must satisfy the Euler-Lagrange equations:

$$\Psi' \left(I_z^2 + \gamma(I_{xz}^2 + I_{yz}^2)(I_x I_z + \gamma(I_{xx} I_{xz} + I_{xy} I_{yz})\right) \tag{2.22}$$

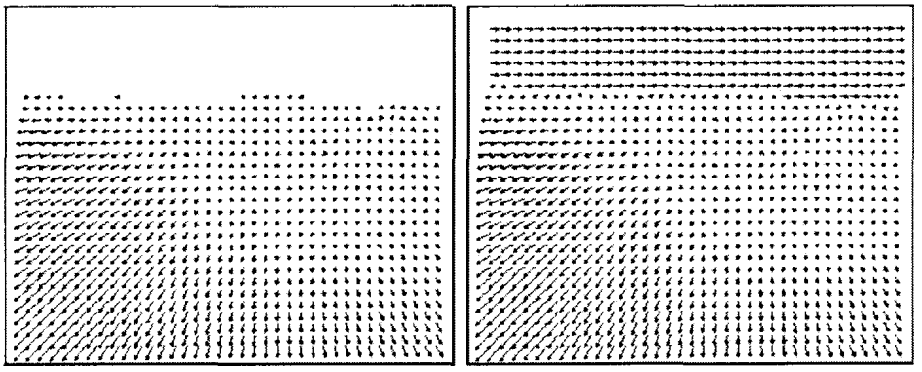
$$- \alpha \mathbf{Div} \Psi'(|\nabla_3 u|^2 + |\nabla_3 v|^2)\nabla_3 u\big) = 0 \tag{2.23}$$

and

$$\Psi' \left(I_z^2 + \gamma(I_{xz}^2 + I_{yz}^2)(I_y I_z + \gamma(I_{yy} I_{yz} + I_{xy} I_{xz})\right) \tag{2.24}$$

$$- \alpha \mathbf{Div} \Psi'(|\nabla_3 u|^2 + |\nabla_3 v|^2)\nabla_3 v = 0\big) \tag{2.25}$$

A consistent numerical scheme based on two data nested fixed point iterations is presented to compute the Euler-Lagrange equations. In order to find the global minimum, a multiscale warping strategy is used: One starts with solving a coarse, smoothed version of the problem. Then the coarse solution is used to warp the second image back into the first. This is done by building an image pyramid with a certain number of levels and a downsampling factor $\eta = 0.95$ from the original image. The computation of velocities begins from the top level, i.e. the coarsest level. Once a solution is obtained, the velocities are projected down to the adjacent finer level, and are used to warp the second image back to the first one. The construction of image pyramid and the projecting of velocities between adjacent levels are performed by bilinear interpolation. If the warping at a level was well performed and the coarse optical flow was good on the previous level, most of the motion between the 2 images will have been removed. Further optical flow calculations and warping as one descends the pyramid can lead to a good final optical flow.



Method	Yosemite with clouds	Yosemite without clouds
2D	$2.46° \pm 7.31°$	$1.59° \pm 1.39°$
3D	$1.94° \pm 6.02°$	$0.98° \pm 1.17°$

Figure 2.3: The computed flow fields for the Yosemite image sequence (left) with and (right) without clouds computed by Brox et al. [6].

2.5 Faisal and Barron 2007 [8]

Faisal and Barron investigated the Brox et al. algorithm in detail and proposed 2 variants on Brox et al.'s algorithm. In the first variant, they adopt standard Horn and Schunck averaging to solve the smoothness term. In the second variant, they employed Brox el al.'s technique, as developed in his Ph.D. thesis, to solve the smoothness term but then used Crammer's rule in an iterative framework rather than using SOR (Successive Over- Relaxation) to obtain convergence. They obtained quantitative error results that were similar and often exceed Brox et al.'s results on the Yosemite sequence.

2.6 3D Optical Flow

3D optical flow can be computed from 3D images (or in the case of scene flow from 2D stereo images). 3D optical flow specifies how much each voxel moves between

adjacent volumes in the dataset 3D optical flow is a simple extension of 2D optical flow [2]. The next sections describe the 3D intensity motion constraint equation as well as the 3D aperture problem.

2.6.1 3D Intensity Motion Constraint Equation

Consider a small 3D $n \times n \times n$ block or voxel at (x, y, z) at time t moving to $(x + \delta x, y + \delta y, z + \delta z)$ at time $t + \delta t$ as shown in Figure 2.4.

Figure 2.4: The small $n \times n \times n$ 3D neighbourhood of voxels centered at (x, y, z) at time t has moved to $(x + \delta x, y + \delta y, z + \delta z)$ at time $t + \delta t$.

We assume a 3D voxel $I(x, y, z)$ at time t moves with a displacement $(\delta x, \delta y, \delta z)$ over time δt. Since $I(x, y, z, t)$ and $I(x + \delta x, y + \delta y, z + \delta z, t + \delta t)$ are equal, we can perform a 1^{st} order Taylor series expansion and obtain similarly to the 2D case:

$$I_x U + I_y V + I_z W = \nabla I \cdot \vec{V} = -I_t,$$

I_x, I_y, I_z and I_t are 3D spatio-temporal intensity derivatives computed via Simoncelli convolution. The 3D velocity is $\vec{V} = (U, V, W)$.

2.6.2 3D Aperture Problem

Equation (2.6.1) describes a plane in 3D space. Any point on that plane is possibly the correct 3D velocity. The velocity on the plane that is closest to the origin is called

the **plane normal** velocity (the velocity normal to a local intensity planar structure) The **line normal** velocity is the velocity on the line caused by the intersection of 2 planes that is closest to the origin. Of course if three planes intersect at a single point, that point is the full 3D velocity. A graphical representation of the 3D plane and line velocities is shown in Figure 2.5 (taken from Spies et al. 2002 [24]). A graphical

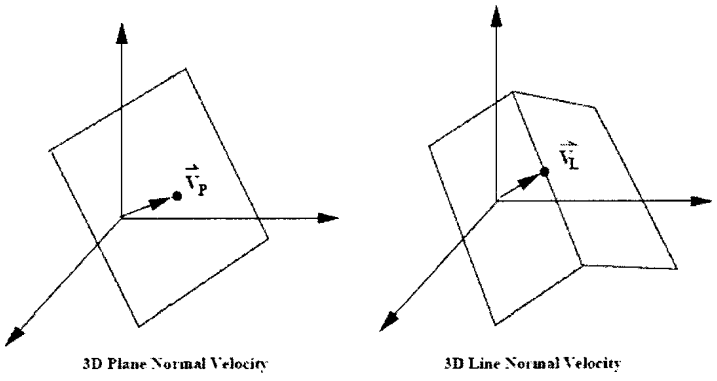


Figure 2.5: The graphical representation of the 3D plane and line velocities.

representation of the types of flows (full, line, and plane) are shown in Figure 2.6 (also taken from Spies et al. 2002 [24]). We are not concerned with the computation

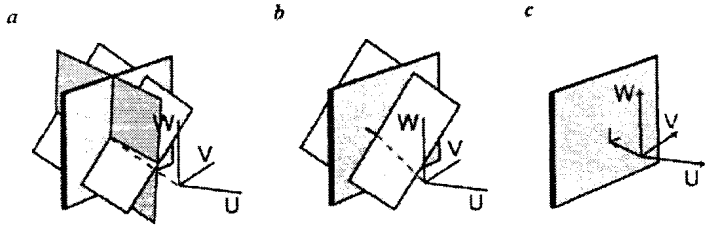


Figure 2.6: Types of Flows: (a) full flow, (b) line normal flow and (c) plane normal flow.

of normal velocities in this thesis.

2.7 3D Horn and Schunck

The extension of Horn and Schunck to 3D is shown in this section (see [2]). 3D Horn and Schunck regularization is written as:

$$F = \int_D (I_x U + I_y V + I_z W + I_t)^2 + \alpha^2 \left(U_x^2 + U_y^2 + U_z^2 + V_x^2 + V_y^2 + V_z^2 + W_x^2 + W_y^2 + W_z^2\right).$$

We use the Euler-Lagrange equations:

$$\begin{aligned} F_U - \frac{d}{dX}F_{U_X} - \frac{d}{dY}F_{U_Y} - \frac{d}{dZ}F_{U_Z} - \frac{d}{dt}F_{U_t} &= 0, \\ F_V - \frac{d}{dX}F_{V_X} - \frac{d}{dY}F_{V_Y} - \frac{d}{dZ}F_{V_Z} - \frac{d}{dt}F_{V_t} &= 0, \\ F_W - \frac{d}{dX}F_{W_X} - \frac{d}{dY}F_{W_Y} - \frac{d}{dZ}F_{W_Z} - \frac{d}{dt}F_{W_t} &= 0. \end{aligned}$$

where:

$$
\begin{aligned}
F_U &= 2I_X(I_X U + I_Y V + I_Z W + I_t) \\
F_V &= 2I_Y(I_X U + I_Y V + I_Z W + I_t) \\
F_W &= 2I_Z(I_X U + I_Y V + I_Z W + I_t) \\
F_{U_X} &= 2\alpha^2 U_X \\
F_{U_Y} &= 2\alpha^2 U_Y \\
F_{U_Z} &= 2\alpha^2 U_Z \\
F_{U_t} &= 2\alpha^2 U_t \\
F_{V_X} &= 2\alpha^2 V_X \\
F_{V_Y} &= 2\alpha^2 V_Y \\
F_{V_Z} &= 2\alpha^2 V_Z \\
F_{V_t} &= 2\alpha^2 V_t \\
F_{W_X} &= 2\alpha^2 W_X \\
F_{W_Y} &= 2\alpha^2 W_Y \\
F_{W_Z} &= 2\alpha^2 W_Z \\
F_{W_t} &= 2\alpha^2 W_t
\end{aligned}
$$

and

$$\begin{aligned}
\frac{dF_{U_X}}{dX} &= 2\alpha^2 U_{XX}, & \frac{dF_{V_Z}}{dZ} &= 2\alpha^2 V_{ZZ} \\
\frac{dF_{U_Y}}{dY} &= 2\alpha^2 U_{YY}, & \frac{dF_{V_t}}{dt} &= 2\alpha^2 V_{tt} \\
\frac{dF_{U_Z}}{dZ} &= 2\alpha^2 U_{ZZ}, & \frac{dF_{W_X}}{dX} &= 2\alpha^2 W_{XX} \\
\frac{dF_{U_t}}{dt} &= 2\alpha^2 U_{tt}, & \frac{dF_{W_Y}}{dY} &= 2\alpha^2 W_{YY} \\
\frac{dF_{V_X}}{dX} &= 2\alpha^2 V_{XX}, & \frac{dF_{W_Z}}{dZ} &= 2\alpha^2 W_{ZZ} \\
\frac{dF_{V_Y}}{dY} &= 2\alpha^2 V_{YY}, & \frac{dF_{W_t}}{dt} &= 2\alpha^2 W_{tt}.
\end{aligned}$$

Since $\nabla^2 U = U_{XX} + U_{YY} + U_{ZZ} + U_{tt}$, $\nabla^2 V = V_{XX} + V_{YY} + V_{ZZ} + V_{tt}$ and $\nabla^2 W = W_{XX} + W_{YY} + W_{ZZ} + W_{tt}$ we can rewrite the Euler-Lagrange equations as:

$$\begin{aligned}
I_X^2 U + I_X I_Y V + I_X I_Z W + I_X I_t &= \alpha^2 \nabla^2 U \\
I_X I_Y U + I_Y^2 V + I_Y I_Z W + I_Y I_t &= \alpha^2 \nabla^2 V \\
I_X I_Z U + I_Y I_Z V + I_Z^2 W + I_Z I_t &= \alpha^2 \nabla^2 W
\end{aligned}$$

Using $\nabla^2 U \approx \bar{U} - U$, $\nabla^2 V \approx \bar{V} - V$ and $\nabla^2 W \approx \bar{W} - W$ we can write:

$$\begin{aligned}
(\alpha^2 + I_X^2) U + I_X I_Y V + I_X I_Z W &= (\alpha^2 \bar{U} + I_X I_t) \\
I_X I_Y U + (\alpha^2 + I_Y^2) V + I_Y I_Z W &= (\alpha^2 \bar{V} + I_Y I_t) \\
I_X I_Z U + I_Y I_Z V + (\alpha^2 + I_Z^2) W &= (\alpha^2 \bar{W} + I_Z I_t).
\end{aligned}$$

The Gauss-Seidel iterative equations can be written as:

$$\begin{aligned} U^{k+1} &= \bar{U}^k - \frac{I_x\left[I_x\bar{U}^k + I_y\bar{V}^k + I_z\bar{W}^k + I_t\right]}{(\alpha^2 + I_x^2 + I_y^2 + I_z^2)}, \\ V^{k+1} &= \bar{V}^k - \frac{I_y\left[I_x\bar{U}^k + I_y\bar{V}^k + I_z\bar{W}^k + I_t\right]}{(\alpha^2 + I_x^2 + I_y^2 + I_z^2)} \quad \text{and} \\ W^{k+1} &= \bar{W}^k - \frac{I_z\left[I_x\bar{U}^k + I_y\bar{V}^k + I_z\bar{W}^k + I_t\right]}{(\alpha^2 + I_x^2 + I_y^2 + I_z^2)}. \end{aligned}$$

$\bar{U}^k$, $\bar{V}^k$ and $\bar{W}^k$ are $n \times n \times n$ averages of neighbourhoods of velocities at iteration k. $\bar{U}^0$, $\bar{V}^0$ and $\bar{W}^0$ are typically set to 0.

2.8 Two Specific Types of 3D Optical Flow: 3D Range Flow and 3D Scene Flow

Range and Scene flow is 3D motion of voxels on a moving surface relative to the sensor (such as 3D motion data of a growing leaf). So, for example, for an observer moving through a stationary environment with a moving car in it, both scene and range flow should have one constant vector for at all background pixels and a second constant vector for all car pixels. This is in contrast to 3D optical flow, which is the 3D motion of each voxel in a volume sequence. Examples of 3D optical flow include the 3D motion vectors computed from gated MRI data of a beating heart or the 3D motion vectors recovered from 3D Doppler radial velocity and reflectivity images. Scene and range flow are distinguished by how they are computed. Range flow uses a 3D depth map of a scene measured over time. These maps can be measured by a range scanner (such as a ShapeGrabber), computed from the disparity map recovered from stereo images or computed from relative depth maps recovered from a monocular motion and structure algorithm. Scene flow, on the other hand, uses disparity and disparity gradients maps (denoted d and p respectively) computed from a stereo sequence,

and the left and right optical flow fields, denoted (u_L, v_L) and (u_R, v_R) respectively, computed from the images in the stereo sequence to compute 3D motion. Note that u and v are the horizontal and vertical components of 2D optical flow. So scene flow does not use any 3D information directly for input.

2.9 Range Flow

There are two basic kinds of range flow problems:

1. motion estimation of locally rigid objects moving in an environment observed by a stationary sensor such as in Spies et al. 2002 [24], Spies et al. 1999 [23], Spies al. 2000 [10], and the regularized range flow algorithm in Spies et al. 2000 [22], and

2. motion estimation of globally rigid objects as in Harville et al. 1999 [12].

2.10 Range Flow Motion Estimation of Globally Rigid Objects

Harville et al. [12] used depth measurements calculated from stereo camera and other sensors along with the traditional linear brightness constraint equations to derive a new depth constraint equation. Consequently, estimation of certain types of motion, such as translation in depth and rotations out of the image plane, becomes more robust. They derive linear brightness and depth change constraint equations that govern the velocity field in 3D for the perspective projection model. They first discuss the classic brightness change constraint equation and its application to motion estimation under perspective camera projection. The coordinate of a point in 3D

space is: $\vec{X} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$ and the 3D velocity is $\vec{V} = \begin{bmatrix} V_x & V_y & V_z \end{bmatrix}^T$. The brightness change constraint equation is $I(x, y, t) = I(x + v_x, y + v_y, t + 1)$ and the depth constraint equation is $Z(x, y, t) + V_z = Z(x + v_x, y + v_y, t + 1)$. This equation is further simplified using a Taylor series expansion and the brightness change constraint equation can be written as $-I_t = \frac{1}{Z}\begin{bmatrix} fI_x & fI_y & -(xI_x + yI_y) \end{bmatrix}\vec{V}^T$. They then introduce and develop a second, analogous constraint that operates directly on depth information. The depth constraint equation is $Z_t = \frac{1}{Z}\begin{bmatrix} fZ_x & fZ_y & -(Z + xZ_x + yZ_y) \end{bmatrix}\vec{V}^T$. They further constraint the 3D world velocities to result from rigid body motion by incorporating a $\mathbf{Q}$ matrix, thus replacing $\vec{V}$ with $\vec{V} = \mathbf{Q}\vec{\phi}$, where

$$\mathbf{Q} = [\boldsymbol{I} - \hat{X}] = \begin{bmatrix} 1 & 0 & 0 & 0 & Z & -Y \\ 0 & 1 & 1 & -Z & 0 & X \\ 0 & 0 & 1 & Y & -X & 0 \end{bmatrix} \tag{2.26}$$

and

$$\hat{X} = \begin{bmatrix} 0 & -Z & Y \\ Z & 0 & -X \\ -Y & X & 0 \end{bmatrix} \tag{2.27}$$

and $\begin{bmatrix} \vec{T}^T & \vec{\Omega}^T \end{bmatrix}^T$. $\vec{\phi} = \vec{T}$ and $\vec{T}$ and $\vec{\Omega}$ are the instantaneous translation and rotation of an environmental object, so $\vec{\phi}$ is the motion parameter vector.

Finally, they show how to combine the brightness change constraint equation and the depth change constraint equation across image pixels into a single linear system which may be solved for the 3-D motion parameters.

They tested the algorithm both on real and synthetic images. They showed results for tracking the pose of faces on both types of images. For each of the two image sequences discussed, motion between all pairs of successive frames were computed using:

1. the brightness change constraint equation (BCCE) only (with measured depth),

2. the depth change constraint equation (DCCE) only, and

3. both constraints together.

Results using either the BCCE or the DCCE alone, with measured depth in each case, were not as good as those obtained using the combined constraints.

2.11 Range Flow Estimation of Locally Rigid Objects

The papers surveyed to understand range flow estimation of locally rigid objects were Spies et al. 2002 [24], Spies et al. 1999 [23], Spies et al. 2000 [10], and the regularized range flow algorithm in Spies et al. 2000 [22] and Barron and Spies 2000 [4]. A discussion of these papers is given below with the least squares equations in Barron and Spies 2000 [4] explained in detail in Chapter 3. Locally rigidity is necessary for the various range flow constraints but global rigidity is not required. Indeed, some of the work of Spies et al 2002 [24] show the range flow on Castor Oil Bean leaves, where global rigidity does not hold.

2.12 Spies et al. 1999 [23]

Spies et al. compute local 3D motion of objects in the scene using a sequence of depth maps that were measured by a Biris laser range sensor. The equation $Z = Z(X, Y, t)$ expresses Z the depth as a function of space and time. Taking the time derivative of Z, they get the equation $\frac{\mathrm{d}Z}{\mathrm{d}t} = \frac{\partial Z}{\partial X}\frac{\mathrm{d}X}{\mathrm{d}t} + \frac{\partial Z}{\partial Y}\frac{\mathrm{d}Y}{\mathrm{d}t} + \frac{\partial Z}{\partial t}$, which leads to the range flow motion constraint equation:

$$Z_X\dot{X} + Z_Y\dot{Y} - \dot{Z} + Z_t = 0 \tag{2.28}$$

Note the dot notation for temporal derivatives, i.e. $\dot{X} = \frac{\partial X}{\partial t}$ and $\dot{X} = \frac{\partial X}{\partial t}$.

The range flow which is defined as the local 3D-motion of objects in the scene is denoted as $f = (\dot{X}, \dot{Y}, \dot{Z}) = (\frac{dZ}{dX}, \frac{dZ}{dY}, \frac{dZ}{dt})$. They solve for the range flow $(\dot{X}, \dot{Y}, \dot{Z})$ via a total least squares method. Equation (2.28) represents a plane in $(\dot{X}, \dot{Y}, \dot{Z})$-space or a plane in 3D velocity space with with surface normal $\begin{bmatrix} Z_X & Z_Y & -1 \end{bmatrix}^T$. One solution, also known as the raw normal flow, is the vector with minimum norm between the origin and the constraint plane and is given by the Equation (2.29) below:

$$f_r = \frac{-Z_t}{Z_X^2 + Z_Y^2 + 1} \begin{bmatrix} Z_X \\ Z_Y \\ 1 \end{bmatrix} \tag{2.29}$$

The raw normal flow is sensitive to noise because it is a local calculation.

2.12.1 The 3D Aperture Problem

Equation (2.28) is one equation with three unknowns, which lead to the 3D aperture problem. The 3D aperture problem means that on a plane, only the movement perpendicular to the plane can be observed. The 3D aperture problem, as manifested by full flow, line flow, and plane flow resulting from the 3D aperture problem were explained above and in two of Spies et al. papers [24, 22].

2.12.2 Solution using Total Least Squares (TLS) Method

One way to solve the range flow problem is to use local TLS. The solution is as follows. First, let $\vec{u} = \begin{bmatrix} \dot{X} & \dot{Y} & -\dot{Z} & 1 \end{bmatrix}^T = \begin{bmatrix} U & V & W & 1 \end{bmatrix}^T$. Then rewrite the range flow motion constraint equation (Equation (2.28)) as $d \cdot u = 0$. If there are N pixels, then there are N such equations. The flow estimation is formulated as:

$$\|D\vec{u}\|_2 \rightarrow min \text{ subject to } \vec{u}^T\vec{u} = 1, \tag{2.30}$$

where $\vec{d} = \begin{bmatrix} Z_X & Z_Y & 1 & Z_t \end{bmatrix}^T$, $\vec{u} = \begin{bmatrix} U & V & W & 1 \end{bmatrix}^T$ and $D = \begin{bmatrix} \vec{d_1} \cdots \vec{dn} \end{bmatrix}^T$. The solution is given by the eigenvector $\hat{e}_4$, corresponding to the smallest eigenvalue λ_4 of the generalized 4×4 structure tensor:

$$J = D^T D = \begin{bmatrix} < Z_X Z_X > & < Z_X Z_Y > & < Z_X > & < Z_X Z_T > \\ < Z_Y Z_X > & < Z_Y Z_Y > & < Z_Y > & < Z_Y Z_T > \\ < Z_X > & < Z_Y > & < 1 > & < Z_T > \\ < Z_T Z_X > & < Z_T Z_Y > & < Z_T > & < Z_T Z_T > \end{bmatrix} \tag{2.31}$$

Here $< \cdot >$ denotes local averaging using a Box or Binomial filter. (Where possible we keep the notation used in the papers we discuss.) The full flow is:

$$\vec{f}_f = \frac{1}{e_{44}} \begin{bmatrix} e_{14} \\ e_{24} \\ e_{34} \end{bmatrix} \tag{2.32}$$

and the eigenvalues and eigenvectors are solved using Jacobi-Rotations transformations. The plane flow is:

$$f_p = \frac{e_{41}}{e_{11}^2 + e_{21}^2 + e_{31}^2} \begin{bmatrix} e_{11} \\ e_{21} \\ e_{31} \end{bmatrix} \tag{2.33}$$

and the line flow is:

$$f_l = \frac{e_{41}}{1 - e_{41}^2 - e_{42}^2} \left[e_{41} \begin{bmatrix} e_{11} \\ e_{21} \\ e_{31} \end{bmatrix} + e_{42} \begin{bmatrix} e_{12} \\ e_{22} \\ e_{32} \end{bmatrix} \right]. \tag{2.34}$$

The performance of their method was assessed on both synthetic and real data. The good quantitative and qualitative data analysis demonstrate the robustness of their method.

2.13 Regularized Range Flow, Spies et al. 2000 [22]

Spies et al. 2000 [22] used range flow to study the motion of a Castor oil bean leaf. The work in Spies et al. 1999 [23] was extended. First, it was shown how to solve range flow via total least squares using Equations (2.28) to (2.34), as was first presented in the paper by Spies et al. 1999 [23]. However, to reduce computational cost, they only compute the range flow where the trace of the tensor matrix is greater than a certain threshold. This can lead to holes in the depth data. So they introduced an iterative regularization approach to fill in these holes and to compute dense range flow. The iterative regularization approach yields the following minimization problem:

$$\int_A \left\{ \omega(Pv - f)^2 + \alpha \sum_{i=1}^{3} (\nabla(v_i)^2 \right\} dr \rightarrow min. \tag{2.35}$$

In Equation (2.35) above, P is the projection matrix which projects onto the subspace determined by the TLS algorithm, ω is the confidence measure of the TLS algorithm, f is the combined plane flow, line flow, and full flow, $v = \begin{bmatrix} U & V & W \end{bmatrix}^T$ and $\nabla = [\partial x, \partial y, \partial t]^T$, which not only considers spatial neighborhoods but also enforces temporal smoothness as well. The confidence measure is used to detect motion discontinuities caused by sharp edges or occlusions and to remove noise, because the TLS algorithm fails when there are motion discontinuities.

Solving for the range flow via the TLS algorithm produces outliers on real data, so Spies et al. introduced another method called direct regularization, which is applied on a sequence of depth maps. The minimization in this case is written as in Equation (2.36) below:

$$\int_A \left\{ (d^T \cdot u)^2 + \alpha \sum_{i=1}^{3} (\nabla(v_i)^2 \right\} dr \rightarrow min. \tag{2.36}$$

Spies et al. tested their algorithm on synthetic and real data and also applied the

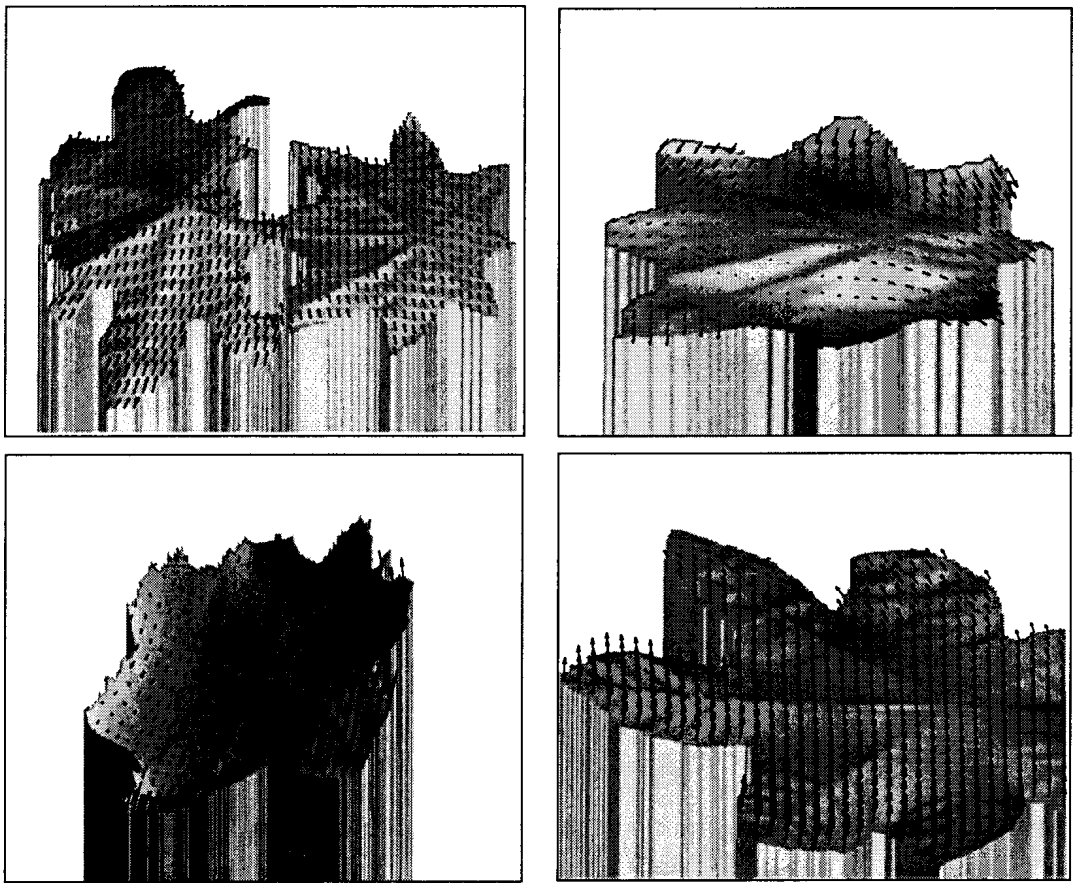


Figure 2.7: Range flow fields for some Castor Oil Bean leaves, from Spies et al. 2002 [24].

range flow algorithm to range data from a growing Castor Oil Bean leaf to study its 3D motion. The regularization algorithm yielded excellent results on pure synthetic data. Most of the error was bias error (the error distinctly was biased in one direction). The remaining error was due to small segmentation mistakes. A range flow field on real data is shown in Figure 2.7. The results show that sub-pixel displacements can be estimated.

2.14 Dense Range Flow from Depth and Intensity Data, Spies et al. 2000 [10]

Spies et al. [10] showed that combining intensity and depth information greatly aids in the estimation of the local 3D range flow of moving surfaces. They demonstrated how that intensity and depth information can be combined in both a local total least squares algorithm and in an iterative global variational technique.

They used two constraint equations, the range flow motion constraint equation

$$Z_X U + Z_Y V + W + Z_t = 0 \tag{2.37}$$

and the brightness change constraint equation

$$I_X U + I_Y V + I_t = 0. \tag{2.38}$$

Again, (total) least squares and regularization solutions are possible. We use both intensity and range data in our range algorithm in Chapter 3.

2.14.1 The Solution Using Total Least Squares

The range flow motion constraint given by Equation (2.37) and the bright change constraint given by Equation (2.38) are recast in a total least squares framework as follows:

$$u_1 Z_X + u_2 Z_Y + u_3 + u_4 Z_t = 0 \quad \text{and} \tag{2.39}$$

$$u_1 I_X + u_2 I_Y + u_4 I_t = 0.. \tag{2.40}$$

Here u_1, u_2, u_3 are the 3D velocities scaled by u_4 as computed by total least squares. The range flow is given by Equation (2.41) below:

$$\begin{bmatrix} U & V & W \end{bmatrix}^T = \frac{1}{u_4} \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^T. \tag{2.41}$$

The vector:

$$\vec{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^T \tag{2.42}$$

is normalized and the following energy functional:

$$J = \int_A [(u_1 Z_X + u_2 Z_Y + u_3 + u_4 Z_t)^2 + \beta^2 (u_1 I_X + u_2 I_Y + u_4 I_t)^2 + \lambda(\vec{u}^T \vec{u})] dX dY \tag{2.43}$$

is minimized. Here, λ is an eigenvalue of the extended structure tensor, which is computed using convolution and smoothing operator. β^2 is a weighting term, weighing the relative importance of the intensity and depth terms.

2.14.2 Solution Using Global Smoothness

To solve for the range flow using the global smoothness method, the following energy functional is minimized:

$$J = \int_A [(u_1 Z_X + u_2 Z_Y + u_3 + u_4 Z_t)^2 + \beta^2 (u_1 I_X + u_2 I_Y + u_4 I_t)^2 + \alpha^2 (\nabla U^2 + \nabla V^2 + \nabla W^2] dX dY. \tag{2.44}$$

Here, the smoothness term is weighed by the constant factor α^2, which specified how important smoothness in the solution. The minimization of J leads to the following

set of linear equations in Equation (2.45) below:

$$\underbrace{\begin{bmatrix} (\alpha^2 + Z_X^2 + \beta^2 I_X^2) + (Z_X Z_Y + \beta^2 I_X I_Y) & Z_X & \\ (Z_X Z_Y + \beta^2 I_X I_Y) & (\alpha^2 + Z_Y^2 + \beta^2 I_Y^2) & Z_Y \\ Z_X & Z_Y & \alpha^2 + 1 \end{bmatrix}}_{=A} \begin{bmatrix} U \\ V \\ W \end{bmatrix} =$$

$$\begin{bmatrix} \alpha^2 \bar{U} - Z_X Z_t - \beta^2 I_X I_t \\ \alpha^2 \bar{V} - Z_Y Z_t - \beta^2 I_Y I_t \\ \alpha^2 \bar{W} - Z_t \end{bmatrix}. \tag{2.45}$$

Spies et al. solve these equations iteratively as:

$$\begin{bmatrix} U^{n+1} \\ V^{n+1} \\ W^{n+1} \end{bmatrix} = A^{-1} \begin{bmatrix} \alpha^2 \bar{U} - Z_X Z_t - \beta^2 I_X I_t \\ \alpha^2 \bar{V} - Z_Y Z_t - \beta^2 I_Y I_t \\ \alpha^2 \bar{W} - Z_t \end{bmatrix}. \tag{2.46}$$

In Equation (2.45), the intensity values are about 100 times larger than the depth values (for the data Spies et al. used) so the intensity values will dominate the numerical calculation (unless appropriately weighted). Therefore, the two constraints are weighted by the factor β^2, where β^2 is $\frac{<\|\nabla Z\|^2>}{<\|\nabla I\|^2>}$. We used a term like this in our range flow algorithm in Chapter 3.

Note that the Biris range sensor collects both range and intensity data under orthographic projection. Therefore, we can assume I_X and I_x and I_Y and I_y are the same (X and Y are 3D coordinates and x and y are their corresponding image coordinates), something not true when the depth data is collected under perspective projection from a stereo calculation. We show how to handle violation of this assumption in Chapter 3.

Spies et al.'s results showed that the integrated use of intensity and depth data greatly improve range flow estimation.

2.15 Barron and Spies 2000

We implemented the paper titles "Quantitative Regularized Range Flow" by Barron and Spies 2000 [4]. We leave the details to Chapter 3.

2.16 Scene Flow

Scene flow was introduced by Vedula et al. 1999 [26]. The two major classes that the scene flow algorithms surveyed can be categorized as:

1. joint motion and disparity estimation methods as in Huguet and Devernay 2007 [14], Rabe et al. 2007 [20], Isard and MacCormick 2006 [15], Zhang et al. 2001 [30] and Patras et al. 1997 [19] and

2. position and velocity estimation steps are decoupled in the algorithm as in the papers Wedel et al. 2008 [29] and in 2010 [28].

2.17 Vedula et al. 1999 nd 2005 [26, 27]

Vedula et al. designed linear algorithms to compute dense 3D scene flow under three different scenarios where the camera is fully calibrated. The first two algorithms compute scene flow from regularized optical flow, assuming the scene shape is known. One of the algorithms uses a single camera, while the other uses a multiple camera setup. The third algorithm computes scene flow from the inconsistencies in multiple optical flows. Vedula et al. 2005 [27] is an improvement to algorithm in Vedula et al. 1999 [26].

If many cameras are used to image the scene, it is impossible to compute scene flow directly from the image measurements without any smoothing or regularization

(except at points with sufficient structure, such as corner points) and only the normal flow can be computed without regularization. This is the aperture problem and means that computation of scene flow is an ill-posed problem. The following equations:

$$\frac{[\mathbf{P}_i]_1(x,y,z)^{\mathrm{T}}}{[\mathbf{P}_i]_3(x,y,z)^{\mathrm{T}}} \tag{2.47}$$

and

$$\frac{[\mathbf{P}_i]_2(x,y,z)^{\mathrm{T}}}{[\mathbf{P}_i]_3(x,y,z)^{\mathrm{T}}} \tag{2.48}$$

where $[\mathbf{P}_i]_j$ is the jth row of $\mathbf{P}_i$, show the relationship between a 3D world point $\mathbf{x} = (x,y,z)^{\mathrm{T}}$ on the surface and 2D image coordinates $\mathbf{u}_i = (u_i, v_i)^{\mathrm{T}}$ of its projection in camera C_i.

Assume that the surface S^t is Lambertian with albedo $\rho = \rho(\mathbf{x};t)$. If the point $\mathbf{x} = (x,y,z)^{\mathrm{T}}$ is visible in camera C_i, the intensity can be written as:

$$I_i^t(\mathbf{u}_i) = -K \cdot \rho(\mathbf{x};t)[\mathbf{n}(\mathbf{x};t) \cdot (\mathbf{r}(\mathbf{x};t)], \tag{2.49}$$

where K is a constant that only depends upon the diameter of the lens and the distance between the lens and the image plane and $\mathbf{n}$ and $\mathbf{r}$ are the scene structure and illumination respectively. Assuming:

1. the illumination is constant or
2. the surface normal does not change rapidly so that $\frac{\mathrm{d}}{\mathrm{d}t}[\mathbf{n} \cdot \mathbf{r}] = 0$,

then the optical flow constraint equation or the gradient constraint equation is:

$$\nabla I_i^t \cdot \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} + \frac{\partial I_i^t}{\partial t} = 0. \tag{2.50}$$

It is not possible to combine the normal flows from several cameras to estimate the scene flow without having to regularize the problem. This is because: if $\mathbf{x} = \mathbf{x}(t)$ is

the 3D path of a point in the world, its instantaneous scene flow is $\frac{\mathrm{d}x}{\mathrm{d}t}$. If the image of this point for camera C_i is $\mathbf{u}_i = \mathbf{u}_i(t)$, then the optical flow is simply the projection of the scene flow into the image plane:

$$\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} = \frac{\partial \mathbf{u}_i}{\partial x}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}, \tag{2.51}$$

where $\frac{\partial \mathbf{u}_i}{\partial x}$ is a 2×3 Jacobian. The gradient constraint equation [Equation (2.50)] can be rewritten as:

$$\nabla I_i^t \cdot \left[\frac{\partial \mathbf{u}_i}{\partial x}\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}\right] + \frac{\partial I_i^t}{\partial t} = 0. \tag{2.52}$$

Differentiating Equation (2.52) with respect to x yields:

$$I_i^t(\mathbf{u}_i) = -K \cdot \nabla\rho(\mathbf{x};t)[\mathbf{n}(\mathbf{x};t) \cdot (\mathbf{r}(\mathbf{x};t)]. \tag{2.53}$$

Because this expression is independent of the camera C_i and depends on properties of the scene (the surface albedo ρ, the scene structure $\mathbf{n}$, and the illumination $\mathbf{r}$), the coefficients of $\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t}$ in Equation (2.52) should ideally always be the same. Hence, any number of copies of the equation will be linearly dependent. Thus, this result means that it is impossible to compute 3D scene flow independently for each point on the object, without some form of regularization of the problem.

Of the three algorithms presented, the first uses a single camera and inverts Equation (2.51) to get:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \frac{\partial \mathbf{x}}{\partial \mathbf{u}_i}\frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} + \frac{\partial \mathbf{x}}{\partial t}. \tag{2.54}$$

In the second algorithm, multiple cameras are used, and the following system of equations is solved:

$$\mathbf{B}\frac{\mathrm{d}\mathbf{x}_j}{\mathrm{d}t} = \mathbf{U}, \tag{2.55}$$

where

$$\mathbf{B} = \begin{bmatrix} \frac{\partial u_1}{\partial x} & \frac{\partial u_1}{\partial y} & \frac{\partial u_1}{\partial z} \\ \frac{\partial v_1}{\partial x} & \frac{\partial v_1}{\partial y} & \frac{\partial v_1}{\partial z} \\ . & . & . \\ . & . & . \\ \frac{\partial u_N}{\partial x} & \frac{\partial u_N}{\partial y} & \frac{\partial u_N}{\partial z} \\ \frac{\partial v_N}{\partial x} & \frac{\partial v_N}{\partial y} & \frac{\partial v_N}{\partial z} \end{bmatrix}, \mathbf{U} = \begin{bmatrix} \frac{\partial u_1}{\partial t} \\ \frac{\partial v_1}{\partial t} \\ . \\ . \\ \frac{\partial u_M}{\partial t} \\ \frac{\partial v_N}{\partial t} \end{bmatrix}. \tag{2.56}$$

In the third case, there are inconsistencies in the multiple optical flows so the solution to Equation (2.51) is written as:

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \left(\frac{\partial \mathbf{u}_i}{\partial \mathbf{x}}\right)^* \frac{\mathrm{d}\mathbf{u}_i}{\mathrm{d}t} + \mu \mathbf{r}_i(\mathbf{u}_i). \tag{2.57}$$

Veluda can only recover the structure where the scene is actually moving. By computing optical flow separately, their algorithms still relies on the accuracy of the optical flow computation. Also, the linear algorithms may be sensitive to noise.

2.18 Joint Motion and Disparity Estimation Scene Flow Methods

The joint motion and disparity estimation methods do the following: at each time instant the two dense motion fields, for the left and the right sequences, and the dense disparity field of the next stereoscopic pair are jointly estimated. Some research using joint motion and disparity estimation technique to calculate scene flow are discussed below.

2.19 Patras et al. 1997 [19]

This work used a joint motion and disparity estimation technique. At time t, they solve the stereo correspondence problem to calculate a smooth disparity field from a pair of stereo images, using a regularization method. The stereo correspondence problem is solved by finding a dense disparity field δ_t at time t, through which every point (x_l, y_l) in the left image is matched to a point (x_l+d, y_l) in the right image. The intensity preservation principle leads to $I_l(x_l, y_l) = I_r(x_l + \delta, y_r)$. They minimize a cost function, which consists of the intensity preservation principle and a smoothness constraint. The kind of smoothness constraint they use is a Discontinuity Adaptive Function, meaning that it regularizes the solution and preserves the discontinuities simultaneously. Preserving discontinuities such as edges is important in order to understand the structure of the scene.

After minimizing the cost function and performing a 1^{st} order Taylor expansion, they devise an iterative solution to solve for the disparity field. To make their solution work for data where large disparity values are present, they use a coarse-to-fine multi-scale method. They also use an error confidence measure to get rid of occlusions (image regions which cannot be seen by both cameras) since occlusions lead to miscalculated disparity fields.

Afterwards, at time $t + 1$, they use the previous disparity field calculate at time t and coarse-to-fine method or pyramid, to jointly estimate the motion field/optical flow at time t and the disparity field at time $t+1$. If there is a correct correspondence between points (i, j) in time t and (i', j') at time $t + 1$, then the following equations hold true:

$$v_r(i', j') = v_l(i, j) \tag{2.58}$$

and

$$\delta_{t+1}(i + u_l, j + v_l) = u_r(i', j') - u_l(i, j) + \delta_t(i, j). \tag{2.59}$$

To solve this step, they minimize a cost function which had 3 major parts. For the first part, the points are not stereo occluded, so the estimation of the disparity field and the estimation of the second disparity field at time t are interconnected. For the other 2 parts, motion fields are estimated independently for the left and right stereo occluded areas respectively. They minimize the cost function and solve it iteratively. This only leads to a partial construction of the disparity field as there were motion occlusions. Therefore, they use error confidence measures to detect the motion occlusions. Then the construction of the disparity field at time $t+1$ is completed by finding the corresponding point $(i+u^l, j+v^l)$ at the left image at time $t+1$ for each point (i,j) in the left image at time t, and assign it to Equation 2.59 .

The advantage of the work of Patras et al. is that due to the joint motion and disparity estimation, the spatial and the temporal consistency of the disparity fields are good and the intermediate pictures natural. One disadvantage is that their algorithm doesn't run in real time. Another disadvantage is that in the joint motion/estimation method proposed in this work the motion occlusion/disclosures and stereo occlusions are detected at steps that follow the estimation of the corresponding fields. In this way the estimation of the disparity and motion fields near these areas is deteriorated, which is shown by their results. Most visible artifacts in the interpolated images appear in such areas. They evaluated their results qualitatively and not quantitatively.

2.20 Zhang et al. 2001 [30]

Zhang et al. provide an energy minimization framework, that includes regularization constraints, to compute dense scene flow.

Initially, they presented a stereo-matching algorithm that employs graph-based image segmentation information to enforce the depth discontinuities. The output of this stereo-matching algorithm includes a disparity map, an occlusion map and a confidence map. Then they show two ways of computing the scene flow.

In the first formulation, they assume that the initial disparity map is accurate enough. The disparity value at point P in frame t in the reference view is denoted as d_t. 3D scene flow at point P is denoted as (u, v, w), where u, v are actually the components of optical flow vector. w is denoted as the change in disparity motion $d_{t+1} - d_t$. Note that w is referred to as d' or p, the gradient of the disparity in other literature. In their algorithm, they assumed that there are $N(C_0, C_1, ...C_{N-1})$ cameras available and the reference camera is C_0. The motion constraints they used combined the optical flow constraints from every single camera. They reprsented the optical flow in camera i as $\frac{\partial I_i}{\partial x}u_i + \frac{\partial I_i}{\partial y}v_i + \frac{\partial I_i}{\partial t} = 0$. Thus for a 2-camera setup, the combined motion constraint for cameras C_0 and C_1 is written in Equation (2.61) as

$$\left(\frac{\partial I_0}{\partial x}|_{(x,y)}u + \frac{\partial I_0}{\partial y}|_{(x,y)}v + \frac{\partial I_0}{\partial t}|_{(x,y)}\right)^2 + \kappa\left(\frac{\partial I_1}{\partial x}|_{(x+d,y)}(u+w) + \frac{\partial I_1}{\partial y}|_{(x+d,y)}v + \frac{\partial I_1}{\partial t}|_{(x+d,y)}\right)^2, \tag{2.60}$$

where κ is the confidence measurement of disparity that is obtained from the stereo matching algorithm, provided that the object point P is visible in camera C_1 or otherwise κ is considered 0. To make the disparity map more accurate, they added the hard constraint ε_h. They added hard constraints by setting the weight of the equation to a very large number for the points with high confidence measure. Equation (2.61) represents the hard constraint at a point in the temporal domain as

$$\varepsilon_h = \mu c(u - \mu_h)^2 + (v - v_h)^2), \tag{2.61}$$

where (u_h, v_h) is the valid motion found by cross-validation. If valid motion has been measured, then μ is the normalized matching score while searching for motion correspondence; and otherwise it is 0 and c is a large constant. To solve for u, v, w,

they minimize the following energy function:

$$\varepsilon_1 = \iint (\varepsilon_m + \varepsilon_h + \varepsilon_s) dxdy \tag{2.62}$$

where ε_m is the motion constraint that combines all the optical flow constraints from all the cameras. ε_s is the smoothness term represented by Equation (2.63) shown below:

$$\varepsilon_s = \gamma \left(\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial y}\right)^2 \right). \tag{2.63}$$

In the second formulation for 3D scene flow computation, they assume that the initial disparity is inaccurate or noisy, and to account for that, they add two more energy terms ε_c and ε_i to the equation. ε_c consists of a stereo constraint and the ε_i made use of confidence map. $\varepsilon_c = -\tau C_{x,y}(d)$ and τ is positive if a point in the corresponding camera is not occluded and otherwise 0. The disparity should maximize the value of $C_{x,y}(d)$, where d is the disparity and $C_{x,y}$ is the correlation volume computed from initial stereo matching. $\varepsilon_i = \zeta(d - d_i)$, where d_i is the initial disparity and ζ is its confidence measurement.

They formulate the problem as computing a 4D vector (u, v, w, d) at every point on the reference image where the initial disparity is used as an initial guess. They use a multi-resolution strategy here as well and minimize the following energy term

$$\varepsilon_1 = \iint (\varepsilon_m + \varepsilon_h + \varepsilon_s + \varepsilon_c + \varepsilon_i) dxdy. \tag{2.64}$$

There are a few differences between formulations one and two. They expect that when large motion exist (i.e. camera ego-motion) formulation 2 will be more appropriate.

One advantage is that the smoothness constraint is not unconditionally applied to the entire image. This is the reason why their method can maintain sharp motion and depth boundaries. The basic advantage of their algorithm was the ability to exploit

all the available constraints in one minimization framework. One disadvantage is that they need to improve the efficiency. They found that if serious occlusions occurred, the results were not good. However, their algorithm maintains very good depth boundaries.

2.21 Isard and MacCormick 2006 [15]

Isard and MacCormick 2006 [15] compute a dense estimate of motion and disparity, given a stereo video sequence containing moving non-rigid objects. This is a dense scene flow technique and it is a fast (real-time) technique but it yields only integer-pixel accurate results. This method estimates motion using loopy belief propagation. Belief propagation is a message passing algorithm for performing inference on graphical models, such as Markov random fields. This algorithm is often used in graphs. The algorithm is sometimes called loopy belief propagation, because graphs typically contain cycles, or loops.

Motion and disparity are estimated simultaneously from a single coherent probabilistic model that correctly accounts for all occlusions, depth discontinuities, and motion discontinuities. The approach models a two-frame stereo and motion problem as a single MRF, and extends to the multi-frame case by using temporal filtering in the same MRF framework. They first describe the MRF employed for two-frame stereo + motion and then they explain the extension to the multi-frame case. Finally, they discuss the use of loopy belief propagation to approximate MAP estimates in these MRFs. The advantages are

1. their estimates are dense,

2. they employ a single coherent probabilistic model, in contrast to iterative segmentation and too much dependence of segmentation give wrong result as in Rabe et al. 2007 [20] and

3. the likelihoods correctly account for occlusions and discontinuities.

The disadvantage is the computational expense.

2.21.1 Isard and MacCormick 2006 Results

Isard and MacCormick found that with the use of temporal filtering, they get better results. Temporal filtering means that the stereo was filtered. Their results also showed that dense stereo and motion produces superior results compared to stereo alone. For example, if there are certain image regions in which stereo alone provides no velocity information, but if stereo and motion does have information in (the majority of) those regions, then better results can be obtained.

2.22 Rabe et al. 2007 [20]

Rabe et al. introduce a fast but sparse scene flow technique. This is another algorithm that fuses optical flow and stereo information together. The performance of this algorithm is real time, but provides sparse results for both the disparity and displacement estimates.

The block diagram in Figure 2.8 shows the main components of the Rabe et al. system. As seen from the Figure 2.8, during each cycle, they obtain a new stereo image pair. First, they track the left image with a Kanade-Lucas-Tomasi tracker to calculate optical flow. Then the left and the right image are sent to the stereo algorithm which gives back depth information. The optical flow along with the depth information is used to calculate the ego-motion. Afterwards, the measurements of the tracking and the stereo module together with the calculated ego-motion are given to the Kalman filter system. For each pixel, one Kalman filter estimates the 6D state vector consisting of the 3D-position and the 3D-motion vector. The 6D state vector

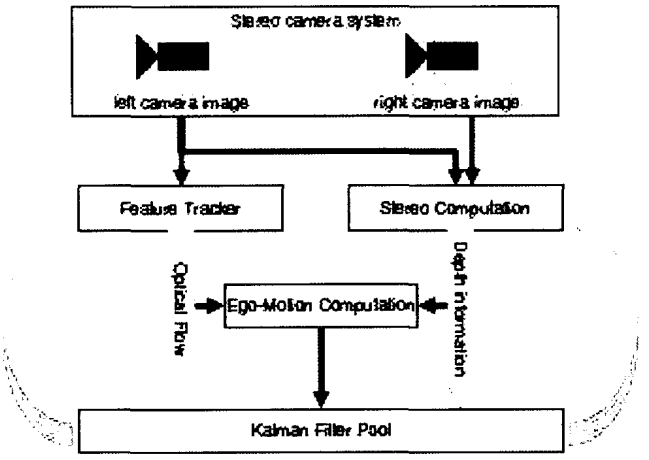


Figure 2.8: Main Components of the System of Rabe et al. 2007 [20]

is shown in Equation (2.65) below:

$$\vec{z} = \begin{bmatrix} \Delta u \\ \Delta v \\ \Delta d \\ s_x \\ s_y \\ s_z \end{bmatrix} = \begin{bmatrix} u_k - u_{k-1} \\ v_k - v_{k-1} \\ d_k - d_{k-1} \\ (c_0 - c_1 s_z^2 \dot{\psi}_{sensor} \\ s_{sensor} sin\theta \\ s_{sensor} cos\theta \end{bmatrix}. \tag{2.65}$$

In this equation, the vector contains the components of the Δu and Δv, which are the measured optical flow and Δd, which is the change of disparity. In addition, the translational velocity components also known as the ego-motion parameters or the 3D motion vectors, s_x, s_y, and s_z, are calculated using the inertial sensor data speed s_{sensor} and the yaw rate $\dot{\psi}_{sensor}$.

In the next cycle, for the next image pair analysis, the already acquired 6D information is used to predict the image position of the pixels in the tracker. This yields

a better tracking performance with respect to speed and robustness. In addition, the predicted depth information is used to improve the stereo calculation.

Rabe et al. showed two ways of calculating the ego-motion. First, they calculate the ego-motion using just the inertial sensors installed in the cars, which just measures the current speed and the yaw rate. This results in wrong 3D motion estimation. Another way of calculating the ego-motion was to use a Kalman filter to accumulate other measurements such as pitch and roll rate, and estimate a state vector containing all ego-motion parameters. This is known as image-based ego-motion compensation. The results are shown in Figure 2.9. The image on the left shows that when ego-motion is computed with just only inertial sensors, the world seems to move downwards as the car undergoes heavy pitch movement and this is the incorrect result. The image on the right shows that they got better results with image-based ego-motion compensation.

Figure 2.9: The results of the Rabe et al. system. The image on the left shows that when ego-motion is computed with only inertial sensors, the world seems to move downwards as the car undergoes heavy pitch movement and this is an incorrect result. The image on the right shows that they got better results with image-based ego-motion compensation.

2.23 Huguet and Devernay 2007 [14]

Huguet and Devernay 2007 address some problems that occur with the reconstruction of scene flow from optical flow in the work of Zhang et al. 2001 [30] and Vedula et

al. 1999 [26]. The reconstruction in these related papers were either under- or over-constrained and the different cameras sometimes yields non-consistent optical flows. To overcome these problems, the algorithm takes into account the epipolar constraint between images taken at the same time, which leads to a minimal parameterization of scene flow from the optical flow and the disparity of a stereo image sequence. Since this parameterization is done in the image space, the problem becomes close to an optical flow estimation problem with more unknown and more measures per image point. Their work is also similar to Brox et al. 2004 [6] as they did not linearize the energy terms minimized by their algorithm. Brox et al. have shown that better results can be obtained by avoiding the linearizaton of the optical flow constraint.

Their goal was to estimate dense scene flow, while preserving the surfaces and motion discontinuities. They first rectify the two image streams so that the stereo disparity was along the horizontal direction in the images. Gaussian smoothing ($\sigma = 1.25$) was also applied to the images in order to avoid numerical instabilities. Their method also benefited from numerical properties of Brox et al. approach: robustness to variation in illumination thanks to the constant image gradient constraints and robustness to stereo or optical flow occlusions, by using the Ψ robust regularization function. Their scene flow algorithm is shown in the Figure 2.10 below and this algorithm is the same diagram in Wedel et al. [29, 28].

The global energy function is shown below:

$$E(u, v, d, d') = E_{Data} + \alpha E_{Smooth}, \tag{2.66}$$

where α is the parameter that controls regularization. The equation for E_{Data} is shown below:

$$E_{Data} = \int_{\Omega} (\beta_{fl} E_{fl} + \beta_{fr} E_{fr} + \beta_{st} E_{st} + \beta_s E_s \mathrm{dx}), \tag{2.67}$$

where β_{fl} is 1 for non occluded pixels for the left optical flow and 0 otherwise and other β functions play a similar role.

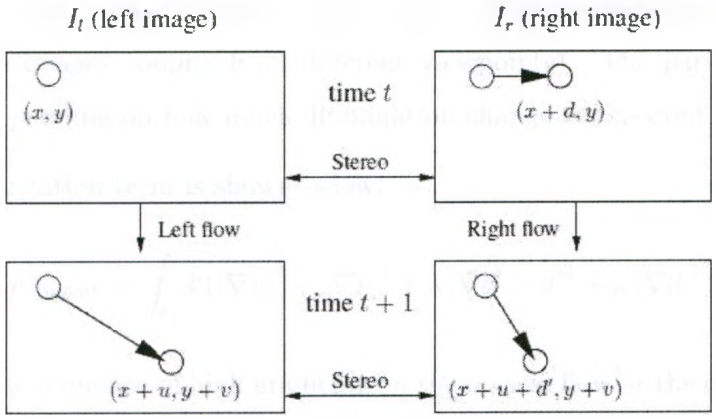


Figure 2.10: Figure taken from Huguet and Devernay 2007 [14], showing the Huguet and Devernay Scene Flow Algorithm.

E_{fl} is the data term corresponding to the left optical flow and E_{fr} corresponds to the right optical flow, which has the same vertical component as the left optical flow. Similarly, E_s corresponds to stereo matching between the left and right images at time t, and E_{st} corresponds to stereo matching at time $t + 1$. E_{fl} and other terms are defined in terms of the equation shown below and Ψ:

$$\Delta(I, \mathrm{x}; I, \mathrm{y}) = |I'(\mathrm{y} - I(\mathrm{x})| + \gamma\,|\nabla I'(\mathrm{y} - \nabla I(\mathrm{x})| \tag{2.68}$$

where the above equation is the difference in intensity and illumination between two image points and $\nabla = (\partial_x, \partial_y)^{\mathrm{T}}$ and $\Psi = \sqrt{(s^2 + \epsilon^2}$, with $\epsilon = 0.01$. The Ψ function is used because pixels in the left image may become occluded in some of the other three images, and quadratic penalisers would give them too much influence on the solution. The Ψ function leads to a robust energy, corresponding to minimization, but is still differentiable everywhere. The Ψ function is applied separately to each data term, since pixels may be occluded by stereo, but not by optical flow, and vice-versa. In addition, Equation (2.68) incorporates a gradient constancy assumption in all data terms, so that the energy is also robust to illumination changes (local or global) and

non-Lambertian surfaces (the stereo terms may be highly affected by such surfaces, since they use images coming from different viewpoints). The parameter γ is set empirically, depending on how much illumination change is expected in the scene.

The regularization term is shown below:

$$E_{Smooth} = \int_{\Omega} \Psi(|\nabla u|^2 + |\nabla v|^2 + \lambda |\nabla d' - d|^2 + \mu |\nabla d|^2). \tag{2.69}$$

By reducing the influence of high gradients on the optical flow or the disparity on the global energy, the Ψ function helps preserving the discontinuities of the functions u, v, d and d'. Unlike in the data term, Ψ is applied to the sum of the gradient norms, since discontinuities usually appear simultaneously in the disparity d, the optical flow (u, v), and the disparity gradient d'.

The problem is ill-posed because the energy is not trivially convex, since the optical flow constraint was not linearized and the nonlinearities are present both in the data term and in the diffusion term of the Euler-Lagrange equations. Therefore, to solve these highly non-linear coupled differential equations, they used an incremental multi-resolution algorithm, with two nested fixed-point iterations (which were obtained by doing a 1^{st} order Taylor expansion of the Euler-Lagrange equations to transform them into a linear system) on the solution for (u, v, d, d') to improve them at each resolution level. The inside fixed point iterations uses the SOR method to solve the final linear system. The stereo image pyramids are computed with a down sampling factor η, $0.5 < \eta < 1$ to get a smooth transition between pyramid levels (they used $\eta = 0.9$). The multi-resolution approach ensures that they converge to a global minimum.

Then the scene flow algorithm is presented. Since the problem is to solve strongly non-linear and non-convex equations, the solution must be carefully initialized in order to avoid local minima which correspond to a wrong solution. In the Brox et al. optical flow algorithm [6], the coarsest or smallest resolution level in the pyramid ensured convergence to the global minimum, since the optical flow is usually small compared

to the image dimensions. However, the amplitude of the stereo disparity is usually comparable to image size (it is usually even bigger than the size of the objects as seen in the images), so they start the scene flow algorithm at an intermediate resolution.

Since the nature of a disparity map is different from the optical flow in the sense that occlusions are larger and the disparity range is comparable to the size of the objects in the image, many difficulties arise from this in multi-resolution stereo algorithms. Therefore, they proposed a two-step algorithm, where the initial solution is bootstrapped by separate solutions to the optical flow and the stereo problem and that initial solution is then refined by their scene flow estimation method.

One advantage of their algorithm is that the method is able to handle real stereo sequences with large motion and stereo discontinuities. Also the regularization terms can handle discontinuities both in the reconstruction and in the motion field, thus allowing fractures to appear on a smooth surface during time. One disadvantage was that it was not computed in real-time. They also want to estimate deterministic continuous function for the β coefficients.

To evaluate their algorithm, they calculated the RMS error. The discontinuity in the mouth area shown in Figure 2.11 was recovered, but it was not subpixel accurate.

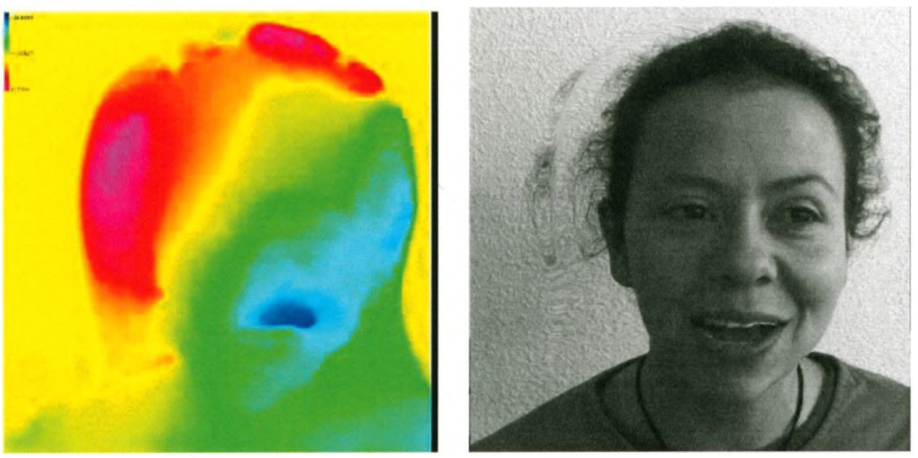


Figure 2.11: Figure taken from Huguet and Devernay 2007 [14], showing that scene flow was recovered in the mouth area.

2.24 Decoupling the Motion and Disparity Estimation Scene Flow Methods

Decoupling the motion and disparity estimation steps in scene flow methods means calculating the disparity and the optical flow separately. Since we implemented Wedel et al.'s algorithm [29, 28] for this thesis the two papers that use this technique are discussed in detail in the next chapter.

Chapter 3

Theoretical Technique

We implemented the scene flow algorithm presented in Wedel et al. 2008 [29], Wedel et al. 2010 [28] and the least squares and regularized range flow technique presented in Barron and Spies 2000 [4]. We explain these range flow and scene flow algorithms in detail in this chapter. The pseudocode of the algorithms is also provided. We conclude the chapter by showing how range flow and scene flow are the same thing mathematically and we also note the basic similarities and differences between the two algorithms.

3.1 Scene Flow

The Wedel et al. 2010 [28] paper is the extended journal version of their 2008 conference paper [29]. We most closely follow the algorithm in the journal paper, rather than the conference paper. Scene flow deals with the recovery 3D observer motion relative to the environment. Scene flow is a combination of optical flow of stereo image sequences and the disparity maps and the disparity gradient maps for those images. Disparity is computed using a stereo algorithm. The stereo algorithm uses epipolar geometry, so that the y pixels in both the left and right images are the same. Thus

disparity needs only to be computed in the x direction (making the stereo algorithm considerably simpler). Stereo epipolar geometry can be achieved by aligning the 2 cameras so that they are frontoparallel (having the same 3D Y and Z coordinates with X in the right camera being $X + b$, where b is the known baseline and having the same direction of gaze).

3.1.1 Wedel et al. Stereo Disparity Estimation

Wedel et al. follow the rectification algorithm given by Liu and Klette 2007 [16] to obtain stereo images that satisfy the epipolar constraint. Given a pair of stereo images, an environmental point $[X, Y, Z]^{\mathrm{T}}$ is projected into cameras images with perspective projection $[x, y]^{\mathrm{T}}$ in the left image and perspective projection $[x + d, y]^{\mathrm{T}}$ in the right image. Equation (3.1) gives the formula:

$$\begin{pmatrix} x \\ y \\ d \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} X f_x \\ Y f_y \\ -b f_x \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \\ 0 \end{pmatrix}, \tag{3.1}$$

where f_x and f_y are the focal lengths (in pixels) for the x and y direction and b (in meters) is the baseline distance between the two camera projection centers. The disparity value d is the difference in the x coordinates of an image correspondence between the left and right image. With known camera intrinsic parameters, the 3D environmental point can easily be recovered from an (x, y, d) measurement. The goal of the stereo correspondence algorithm is to estimate the disparity d, for every non-occluded pixel in the left image. This can be accomplished by local methods (using a small matching window from the left image to the right image) or global methods (incorporating some global energy minimization). Two such methods will be outlined in the next chapter.

3.1.2 The Outline of the Wedel et al. Scene Flow Algorithm

The Figure 3.1 below shows the outline of their scene flow algorithm. This figure describes how scene flow can be calculated from two consecutive images.

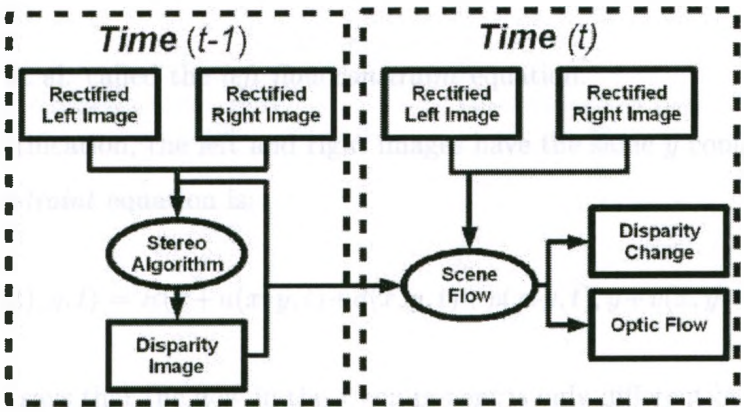


Figure 3.1: The outline of the scene flow algorithm, taken from Wedel et al. 2010 [28]. From the figure, it is evident that the information needed to compute scene flow at time $t-1$ are the rectified stereo image pairs at times $t-1$ and t, along with the disparity map at time $t-1$.

As can be seen from Figure 3.1, the rectified stereo images pair is required for both the previous and current time frame. The rectified stereo image pair in the previous time frame is passed to the stereo algorithm and the stereo algorithm produces the disparity image. The disparity image, d, and the rectified left and right images from both times frames are then passed to the scene flow algorithm to produce left and right optical flow fields ((u_L, v_L) and (u_R, v_R)) and the disparity gradient field, p, between the image pairs. Given left optical flow, (u_L, v_L) and d and p, the 3D velocity at each pixel (x, y) can be computed (we give these equations below).

3.1.3 Wedel et al. Constraint Equations

Several equations arise because of the "consistent intensity assumption", which means that the image intensity is the same in both images for the same 3D environmental

point in the scene. If $L(x, y, t)$ is the intensity of the left image at pixel position $[x, y]^{\mathrm{T}}$ and time t, then

$$L(x, y, t) = L(x + u(x, y, t), y + v(x, y, t), t + 1), \tag{3.2}$$

which Wedel et al. called the *left flow constraint* equation.

Due to rectification, the left and right images have the same y component so the *right flow constraint* equation is:

$$R(x+d(x, y, t), y, t) = R(x+u(x, y, t)+d(x, y, t)+p(x, y, t), y+v(x, y, t), t+1). \tag{3.3}$$

This equation says that the flow in the x component is only different by the disparity d and the disparity change p. Also, p (the disparity change) is defined as $u^R - u^L$, which are the respective left and right optical flows.

Moreover, the grayvalue of the left and right images should also be the same. This leads to the 3^{rd} constraint equation shown below in Equation (3.4):

$$L(x + u(x, y, t), y + v(x, y, t), t + 1) =$$

$$R(x + u(x, y, t) + d(x, y, t) + p(x, y, t), y + v(x, y, t), t + 1). \tag{3.4}$$

The motion and the disparity constraints employed by the scene flow algorithm is illustrated in Figure 3.2 below.

After rearranging Equations (3.2), (3.3) and (3.4), we get the following 3 equations

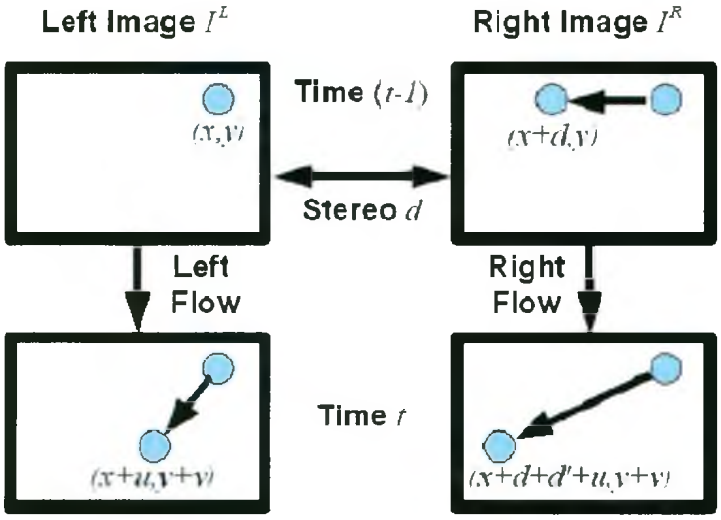


Figure 3.2: This figure, taken from Wedel et al. [28], shows the motion and disparity constraints of the scene flow algorithm. The intensities of the corresponding points in the left and right stereo images and in successive time frames are assumed to be constant.

shown below in Equations (3.5) to (3.7):

$$E_{LF} := L(x+u, y+v, t+1) - L(x, y, t) = 0, \tag{3.5}$$

$$E_{RF} := R(x+d+p+u, y+v, t+1) - R(x+d, y, t) = 0 \text{ and} \tag{3.6}$$

$$E_{DF} := R(x+d+p+u, y+v, t+1) - L(x+u, y+v, t+1) = 0. \tag{3.7}$$

The relationships of E_{LF}, E_{RF} and E_{DF} are illustrated in Figure 3.3 below.

3.1.4 Wedel et al. Energy Equations

The scene flow algorithm estimates u, v and p by minimizing an energy functional consisting of a data term (derived from constraints) and a smoothness term that

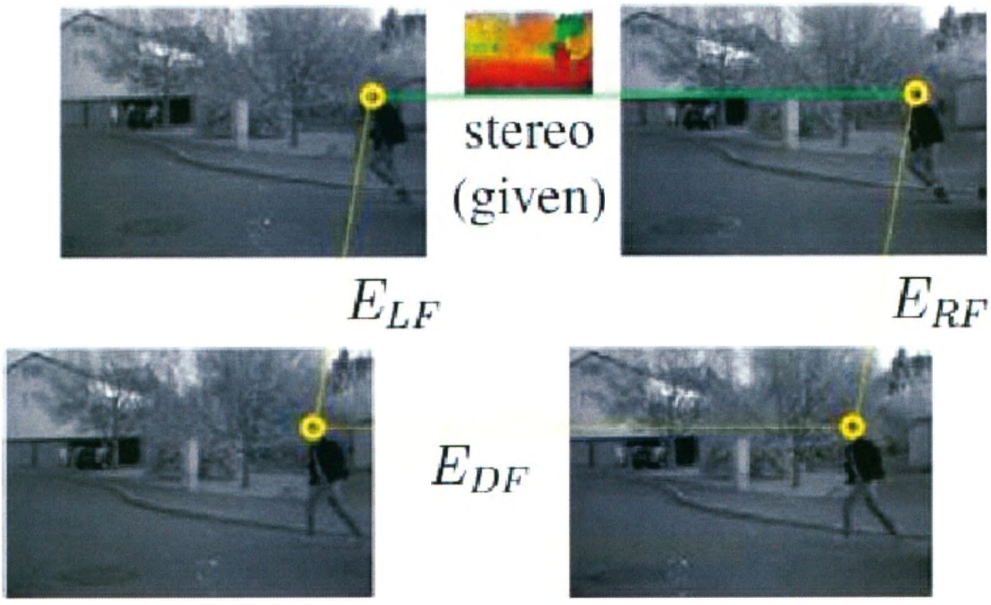


Figure 3.3: This figure, taken from Wedel et al. 2010 [28], illustrates the relationships between the quantities in Equations (3.5) to (3.7) for two stereo image pairs.

enforces smooth and dense scene flow parameters:

$$E_{SF} = \int_{\Omega} (E_D + E_S) dxdy, \tag{3.8}$$

where E_D and E_S are the data and smoothness term and E_{SF} is the scene flow. By using the constraint equations in Equations (3.5) to (3.7), we get the following data term in Equation (3.9):

$$E_D = \int_{\Omega} \Psi E_{LF}^2 + c(x,y,t)\Psi E_{RF}^2 + c(x,y,t)\Psi E_{DF}^2 dxdy. \tag{3.9}$$

In Equation (3.9), $\Psi(s)^2 = \sqrt{s^2 + \epsilon}$, where ϵ=0.0001, and $\Psi(s)^2$ denotes a robust function that compensates for outliers and the function $c(x, y, t)$ returns 0 if there is no disparity known at $[x, y]^{\mathrm{T}}$ and returns 1 where a disparity exists at $[x, y]^{\mathrm{T}}$. The disparity might not be known at a location because of occlusion or a sparse stereo method.

Moreover, local deviations in the scene flow components are penalized by the smoothness term and the smoothness term employs the same robust function as the data term in order to deal with discontinuities in the scene flow field:

$$E_S = \Psi(\lambda \left|\nabla u\right|^2 + \lambda \left|\nabla v\right|^2 + \gamma \left|\nabla p\right|^2). \tag{3.10}$$

where $\nabla = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right]^{\mathrm{T}}$. The parameters λ and γ regulate the importance of the smoothness constraint, optical flow and disparity gradient values respectively.

3.1.5 The Minimization of the Energy Equations

This section shows the minimization of the energy equations found by computing the Euler-Lagrange equations.

3.1.5.1 The Initial Euler-Lagrange Equations

The minimization of energy uses the Euler-Lagrange equations:

$$\Psi' E_{LF}^2 L_x + c\Psi' E_{RF}^2 R_x + c\Psi' E_{DF}^2 E_{DF}(R_x - L_x) - \lambda div(\nabla u E_s{}') = 0, \tag{3.11}$$

$$\Psi' E_{LF}^2 L_y + c\Psi' E_{RF}^2 R_y + c\Psi' E_{DF}^2 E_{DF}(R_y - L_y) - \lambda div(\nabla v E_s{}') = 0, \tag{3.12}$$

and

$$c\Psi' E_{RF}^2 E_{RF} R_x + c\Psi' E_{DF}^2 E_{DF} R_x - \lambda div(\nabla p E_s{}') = 0, \tag{3.13}$$

with

$$E'_s := \Psi'(\lambda \left|\nabla u\right|^2 + \lambda \left|\nabla v\right|^2 + \gamma \left|\nabla p\right|^2). \tag{3.14}$$

where $\Psi'(s^2) =$ denotes the derivatives of Ψ with respect to s^2. Partial derivatives of $R(x+u, y+v, t)$ and $L(x+d+p+u, y+v, t)$ are denoted by subscripts x and y. For simplicity, $c = c(x, y, t)$. We can also see from Equations (3.10) and (3.14) that $E'_s = E_s$ with all Ψ functionals being replaced by Ψ'.

3.1.5.2 Warping and Linearizaton

Because the Euler-Lagrange equations are non-linear in the unknowns $[u, v, p]^{\mathrm{T}}$, they used the strategy of two nested fixed point iterations loops as suggested by Brox et al. 2004 [6]. The outer fixed point iteration loop contains the linearizaton of the E_{LF}, E_{RF}, and E_{DF}. They start with $[u^0, v^0, p^0]^{\mathrm{T}} = [0, 0, 0]^{\mathrm{T}} =$ for all $[x, y]^{\mathrm{T}}$, where k is the increment of the unknowns $[\delta u^0, \delta v^0, \delta p^0]^{\mathrm{T}}$ that is estimated in each iteration. The second image is then warped according to a new estimate shown below:

$$[u_{k+1}, v_{k+1}, p_{k+}]^T = [u_k + \delta u_k, v_k + \delta v_k, p_k + \delta p_k]^T. \tag{3.15}$$

The linearizaton is performed in the left and right images as follows:

$$L(x + u^{k+1}, y + v^{k+1}, t) \approx L(x + u^k, y + v^k, t) + \delta u^k L_x + \delta v^k L_y \tag{3.16}$$

and

$$R(x + d + (p)^{k+1} + u^{k+1}, y + v^{k+1}, t) \approx R(x + d + (p)^k + u^k, y + v^k, t) +$$

$$\delta u^k R_{(x+d)} + \delta (p)^k R_{(x+d)} + \delta v^k R_y. \tag{3.17}$$

In Equation (3.17), $R(x+d)$ is the derivative of R with respect to $(x+d)$. From Equations (3.16) and (3.17), we can derive linearized versions of E_{LF}, E_{RF}, and E_{DF}.

The warping is combined with a pyramid or coarse-to-fine strategy, meaning that they work with the down-sampled versions of the image whose resolutions are successively refined with each iteration.

3.1.5.3 Final Linearized Euler-Lagrange Equations

The remaining non-linearity in the Euler-Lagrange equations is because of the robust function, Ψ. In the inner fixed point loop, the Ψ' expressions are kept constant and are recomputed after each iteration l. This leads to the following linear equations:

$$\begin{aligned}
&\Psi'^{k,l}_{LF} \cdot (E^k_{LF} + L^k_x \delta u^{k,l+1} + L^k_y \delta v^{k,l+1}) L^k_x \\
&+c\Psi'^{k,l}_{RF} \cdot (E^k_{RF} + R^k_x(\delta u^{k,l+1} + \delta d'^{(k,l+1)}) + R^k_y \delta v^{k,l+1}) R^k_x + \\
&\lambda div(\Psi'^{k,l}_S \cdot \nabla(u^k + \delta u^{k,l+1})) = 0, \qquad (3.18) \\
&\Psi'^{k,l}_{LF} \cdot (E^k_{LF} + L^k_x \delta u^{k,l+1} + L^k_y \delta v^{k,l+1}) L^k_y \\
&+c\Psi'^{k,l}_{RF} \cdot (E^k_{RF} + R^k_x(\delta u^{k,l+1} + \delta d'^{(k,l+1)}) + R^k_y \delta v^{k,l+1}) R^k_y + \\
&\lambda div(\Psi'^{k,l}_S \cdot \nabla(v^k + \delta v^{k,l+1})) = 0 \qquad (3.19)
\end{aligned}$$

and

$$\begin{aligned}
&c\Psi'^{k,l}_{RF} \cdot (E^k_{RF} + R^k_x(\delta u^{k,l+1} + \delta d'^{(k,l+1)}) + R^k_y \delta v^{k,l+1}) R^k_x \\
&+c\Psi'^{k,l}_{DF} \cdot (E^k_{DF} + R^k_x \delta d'^{(k,l+1)}) R^k_x - \\
&\gamma div(\Psi'^{k,l}_S \cdot \nabla(p^k + \delta p^{(k,l+1)})) = 0, \qquad (3.20)
\end{aligned}$$

with

$$\Psi'^{k,l}_* := \Psi(E_*(u^k + \delta u^{k,l}, v^k + \delta v^{k,l}, d'^k + \delta d'^{k,l})). \qquad (3.21)$$

These equations are solved using SOR (successive over-relaxation is an iterative method to solve large systems of equations).

3.1.6 Derivation of Image Scene Flow to 3D Scene Flow (World Flow)

After deriving the image scene flow as a combined estimation of optical flow and disparity change, Wedel et al. use the image scene flow to compute world point tuples that represent the start and end point of the 3D scene flow. These 3D environmental points can be computed as:

$$X_{t-1} = (x - x_0)\frac{b}{d}, \quad Y_{t-1} = (y - y_0)\frac{f_y}{f_x}\frac{b}{d} \quad \text{and} \quad Z_{t-1} = \frac{f_x b}{d}, \tag{3.22}$$

and

$$X_t = (x + u - x_0)\frac{b}{d+p}, \quad Y_t = (y + u - y_0)\frac{f_y}{f_x}\frac{b}{d+p} \quad \text{and} \quad Z_t = \frac{f_x b}{d+p}, \tag{3.23}$$

where (x_0, y_0) are offsets computed so that the image origin is at the center of the flow field.

The 3D scene flow $[U, V, W]^T = [X', Y', Z']^{\mathrm{T}}$ is the subtraction between these two 3D environmental points:

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} X_t - X_{t-1} \\ Y_t - Y_{t-1} \\ Z_t - Z_{t-1} \end{pmatrix} = b \begin{pmatrix} \frac{x-x_0}{d} - \frac{x+u-x_0}{d+p} \\ \frac{y-y_0}{d} - \frac{y+v-y_0}{d+p} \\ \frac{f_x}{d} - \frac{f_x}{d+p}. \end{pmatrix} \tag{3.24}$$

3.1.7 Wedel et al. 2008 Results

Wedel et al. minimized the energy functional using the Euler-Lagrange equations and solved for (u, v, p). Then they tested their scene flow algorithm with 4 stereo algorithms: semi-global matching (SGM) and SGM with hole filling (which favors

smaller disparities), correlation pyramid stereo and an integer accurate census-based stereo algorithm. They achieved lower scene flow errors than Huguet et al. method [14]. They also found that the RMS (root mean square) error of their scene flow is much smaller. RMS measures the magnitude of a varying quantity. RMS is an useful statistical measure when the variates are positive and negative such as in sinusoid data. They also observed that SGM with hole filling yielded inferior results than the other stereo methods and that normal SGM yielded the best results.

Stereo Algorithm	RMS_d (density)	Without occluded areas			With occluded areas		
		$RMS_{u,v}$	$RMS_{u,v,d'}$	$AAE_{u,v}$	$RMS_{u,v}$	$RMS_{u,v,d'}$	$AAE_{u,v}$
Huguet *et al.* [7]	3.8 (100%)	0.37	0.83	1.24	0.69	2.51	1.75
Flow Only	$d' = 0$ for	0.34	1.46	1.26	0.67	2.85	1.72
Flow Only*	evaluation	0.30	1.46	0.95	0.64	2.85	1.36
Ground truth		0.33	0.58	1.25	0.67	2.40	1.78
Ground truth*		0.31	0.56	0.91	0.65	2.40	1.40
SGM [5]	2.9 (87%)	0.35	0.64	1.33	0.66	2.45	1.82
SGM*		0.34	0.63	1.04	0.66	2.45	1.50
Fill-SGM	10.9 (100%)	0.43	0.75	2.18	0.77	2.55	2.99
Fill-SGM*		0.45	0.76	1.99	0.77	2.55	2.76
Correlation [3]	2.6 (43%)	0.34	0.75	1.31	0.67	2.51	1.84
Correlation*		0.33	0.73	1.02	0.65	2.50	1.52
Census based [15]	7.8 (16%)	0.36	1.08	1.30	0.67	2.65	1.75
Census based*		0.32	1.14	1.01	0.65	2.68	1.43

Figure 3.4: Figure taken from Wedel et al. 2008 [29] showing the evaluation of their algorithm.

3.1.8 Wedel et al. 2010 Results

Wedel et al. used RMS (root mean square) error to evaluate their results in this paper. Some root mean square errors (in pixels) and average angular errors in degrees for computed scene flow for a rotating sphere sequence are shown in Figure 3.5.

3.2 Pseudo code of Scene Flow Algorithm

We implemented the scene flow algorithm according to the pseudo code that was provided in Wedel et al. 2010 [28] as shown below.

Stereo Algorithm	RMS_d (density)	Without occluded areas			With occluded areas		
		$RMS_{u,v}$	$RMS_{u,v,d'}$	$AAE_{u,v}$	$RMS_{u,v}$	$RMS_{u,v,d'}$	$AAE_{u,v}$
Ground truth	0 (100%)	0.31	0.56	0.91	0.65	2.40	1.40
SGM [6]	2.9 (87%)	0.34	0.63	1.04	0.66	2.45	1.50
Correlation [4]	2.6 (43%)	0.33	0.73	1.02	0.65	2.50	1.52
Census based [17]	7.8 (16%)	0.32	1.14	1.01	0.65	2.68	1.43
Hug.-Dev. [8]	3.8 (100%)	0.37	0.83	1.24	0.69	2.51	1.75
Fill-SGM	10.9 (100%)	0.45	0.76	1.99	0.77	2.55	2.76

Figure 3.5: Figure taken from Wedel et al. 2010 [28] showing the evaluation results in their IJCV paper.

for all levels **do**
  for outer iterations **do**
    Compute structure (Algorithm 2)
    $u_{tmp} = u$; $v_{tmp} = v$; $dv1 = 0$; $p_{tmp} = d'$.
    for all inner iterations do **do**
      Build Equation System (Algorithm 3)
      Compute Diffusivity (Algorithm 4)
      for all SOR iterations **do**
        SOR step (Algorithm 5)
      end for
    end for
    warp $L(x, y, t)$ and $R(x + d, y, t)$ using u, v and p.
  end for
  warp u, v and p to upper level (double size and interpolate).
end for

Note that line 14 (warp $L(x, y, t)$ and $R(x + d, y, t)$ using u, v and p) is unclear. We interpret this to mean warp $L(x, y, t+1)$ and $R(x+d, y, t+1)$ back into $L(x, y, t)$ and $R(x + d, y, t)$ using u, v and p. Then after the highest level of the pyramid has been processed this warping removed that motion from the images. Then processing at the second highest to lowest levels in the pyramid only computes u, v and p for the first left and right images and their warped versions. u_{tmp}, v_{tmp} and p_{tmp}, besides

being used in Wedel et al.'s "trick" to avoid an explicit calculation of δu, δv and δp and thus speed up the calculation is also used to retain the velocities used to compute the warped images. We contacted Wedel by email about this and later found that, they do all of their warping in the **compute structure** algorithm.

Moreover, Wedel wrote[1] that "he didn't warp the images, but only warped the derivative images instead," which is done inside Algorithm 2, called **compute structure** such as in line 1 $[L_x\ L_y]^T = \frac{1}{2}(\nabla L(x+u, y+v, t+1) + \nabla L(x, y, t))$. So if only the derivative images are warped, we believe that line 14 of Algorithm 1, where it talks about warping is not in the right place.

Algorithm 2, called **compute structure** is shown below:

for all pixels **do**

$[L_x\ L_y]^T = \frac{1}{2}(\nabla L(x+u, y+v, t+1) + \nabla L(x, y, t))$

$L_t = L(x+u, y+v, t+1) - L(x, y, t)$

if disparity d is known, i.e. c(x,u)=1 **then**

$[R_x\ R_y]^T = \frac{1}{2}(\nabla R(x+u+d+p, y+v, t+1) + \nabla R(x+d, y, t))$

$R_t = R(x+u+d+p, y+v, t+1) - R(x, y, t)$

$D_x = \frac{1}{2}(\frac{\partial}{\partial x} R(x+u+d+p, y+v, t+1) + \frac{\partial}{\partial x} L(x+d, y+v, t))$

$D_t = R(x+u+d+p, y+v, t+1) - L(x+u, y+v, t+1)$

else

$[R_x\ R_y]^T = 0$

$D_x = 0,\ R_t = 0,\ D_t = 0$

end if

end for

Algorithm 3, called **build equation system** is shown below:

for all pixels **do**

$E_{LF} = L_t + (u_{\text{tmp}} - u)L_x + (v_{\text{tmp}} - v)L_y$

$\Psi'_{LF} = \frac{1}{\sqrt{E_{LF}^2 + \epsilon^2}}$

$E_{RF} = R_t + (u_{\text{tmp}} + p_{\text{tmp}} - u - p)R_x + (v_{\text{tmp}} - v)R_y$

[1]By email on June 4th,2011.

$\Psi'_{RF} = \frac{1}{\sqrt{E^2_{RF}+\varepsilon^2}}$

$E_{DF} = D_t(p_{\text{tmp}} - p)D_x$

$\Psi'_{DF} = \frac{1}{\sqrt{E^2_{DF}+\varepsilon^2}}$

$A_{uu} = \Psi'_{LF}L_xL_x + \Psi'_{RF}R_xR_x$

$A_{uv} = \Psi'_{LF}L_xL_y + \Psi'_{RF}R_xR_y$

$A_{vv} = \Psi'_{LF}L_yL_y + \Psi'_{RF}R_yR_y$

$A_{up} = \Psi'_{RF}R_xR_x$

$A_{vp} = \Psi'_{RF}R_xR_y$

$A_{pp} = \Psi'_{RF}R_xR_x + \Psi'_{DF}D_xD_x$

$b_u = \Psi'_{LF}L_x(L_t + u_{\text{tmp}}L_x + v_{\text{tmp}}L_y) + \Psi'_{RF}R_x(R_t + (u_{\text{tmp}} + p_{\text{tmp}})R_x + v_{\text{tmp}}R_y)$

$b_v = \Psi'_{LF}L_y(L_t + u_{\text{tmp}}L_x + v_{\text{tmp}}L_y) + \Psi'_{RF}R_y(R_t + (u_{\text{tmp}} + p_{\text{tmp}})R_x + v_{\text{tmp}}R_y)$

$b_p = \Psi'_{RF}R_x(R_t + (u_{\text{tmp}} + p_{\text{tmp}})R_x + v_{\text{tmp}}R_y) + \Psi'_{DF}D_x(D_t + p_{\text{tmp}}D_x)$

end for

Algorithm 4, called **computing diffusivity** is shown below:

for all pixels **do**

for all $\alpha\epsilon\{u, v, p\}$ **do**

$R_{\alpha,north} = (\alpha(x,y) - \alpha(x,y-1))^2 + \frac{1}{16}(\alpha(x+1,y) - \alpha(x-1,y) + \alpha(x+1,y-1) - \alpha(x-1,y-1))^2$

$R_{\alpha,east} = (\alpha(x,y) - \alpha(x-1,y))^2 + \frac{1}{16}(\alpha(x,y+1) - \alpha(x,y-1) + \alpha(x-1,y+1) - \alpha(x-1,y-1))^2$

$R_{\alpha,south} = (\alpha(x,y+1) - \alpha(x,y))^2 + \frac{1}{16}(\alpha(x+1,y+1) - \alpha(x-1,y+1) + \alpha(x+1,y) - \alpha(x-1,y))^2$

$R_{\alpha,west} = (\alpha(x+1,y) - \alpha(x,y))^2 + \frac{1}{16}(\alpha(x+1,y+1) - \alpha(x+1,y-1) + \alpha(x,y+1) - \alpha(x,y-1))^2$

end for

for all dir ϵ $\{north, east, south, west\}$ **do**

$R_{\text{dir}} = \frac{\lambda}{\sqrt{R_{u,\text{dir}} + R_{v,\text{dir}} + \frac{\lambda^2}{\gamma^2}R_{p,\text{dir}}}}$

end for

end for

Note that to prevent the denominator in $R_{\text{dir}} = \frac{\lambda}{\sqrt{R_{u,\text{dir}}+R_{v,\text{dir}}+\frac{\lambda^2}{\gamma^2}R_{p,\text{dir}}}}$ from vanishing (which would make R_{dir} infinity), we added $\epsilon = 0.001$ as suggested by Wedel. [2] Algorithm 5 for the **SOR step** is shown below:

for all pixels **do**

$R_{\text{sum}} = R_{\text{north}} + R_{\text{south}} + R_{\text{west}} + R_{\text{east}} + \varepsilon^2$

$u_R = R_{\text{north}}u(x, y-1) + R_{\text{south}}u(x, y+1) + R_{\text{west}}u(x-1, y) + R_{\text{east}}u(x+1, y)$

$u(x, y) = (1-\omega)u(x, y) + \frac{\omega}{A_{uu}+R_{\text{sum}}}(u_R - b_u - A_{uv}v(x, y) - A_{up}p(x, y))$

$v_R = R_{\text{north}}v(x, y-1) + R_{\text{south}}v(x, y+1) + R_{\text{west}}v(x-1, y) + R_{\text{east}}v(x+1, y)$

$v(x, y) = (1-\omega)v(x, y) + \frac{\omega}{A_{vv}+R_{\text{sum}}}(u_R - b_v - A_{uv}u(x, y) - A_{vp}p(x, y))$

$p_R = R_{\text{north}}p(x, y-1) + R_{\text{south}}p(x, y+1) + R_{\text{west}}p(x-1, y) + R_{\text{east}}p(x+1, y)$

$p(x, y) = (1-\omega)p(x, y) + \frac{\omega}{A_{pp}+R_{\text{sum}}}(u_R - b_p - A_{up}u(x, y) - A_{vp}p(x, y))$

end for

Note that we changed the terms $(v_R - b_v - A_{uv}u(x, y) - A_{vp}p(x, y))$ and $(p_R b_p - A_{up}u(x, y) - A_{vp}p(x, y))$ in the **SOR step** pseudo code that initially were $(u_R - b_u - A_{up}u(x, y) - A_{vp}p(x, y))$ and $(u_R - b_u - A_{up}u(x, y) - A_{vp}p(x, y))$ in Wedel et al.'s IJCV paper [28] (lines 6 and 8 in algorithm 5) These corrections have been verified as correct with Wedel by email.

3.3 Range Flow

Range flow requires 2 depth maps, Z_t and Z_{t+1}. Using the range flow constraint equation, $[U, V, W]^T$ can be computed directly from the Z values and their 1^{st} order derivatives. Z can be computed from stereo image sequences (as has been done for Wedel et al.'s scene flow), from Z data measured directly from a laser scanner or from relative/absolute depth computed by monocular/binocular motion and structure algorithms. The algorithms below were initially based on being able to measure Z data via a laser scanner assuming orthographic projection. Since we compute

[2] By email on July 5th,2011.

our depth data from disparity data computed from stereo images under perspective projection we need to make a slight modification to these algorithms that we describe in the next section.

In the following sections, we discuss some of the range flow algorithms presented in Spies 2000l. [4, 10, 22, 24]. We describe the range flow algorithms as we implemented them.

3.3.1 Least Squares Range Flow

Least Squares Range Flow was introduced in the paper "Differential Range Flow" by Spies et al. 1999 [23]. The motion constraint equation can easily be extended into 3D using range derivatives as:

$$Z_X U + Z_Y V + Z_Z W + Z_t = 0. \tag{3.25}$$

For range data $Z_Z = 1$ (we only have Z values at environmental surfaces) and the range constraint equation becomes:

$$Z_X U + Z_Y V + W + Z_t = 0. \tag{3.26}$$

Sometime $+W$ is written as $-W$ in this equation, depending on whether a 3D left-handed or right-handed coordinate system is being used (we are using a left handed system, as Z gets bigger as we move further into the environment). Note that variables X and Y are 3D coordinates. Under orthographic projection (which a range sensor approximates) $Z_X \approx Z_x$ and $Z_Y \approx Z_y$, where x and y are the image coordinates (pixel coordinates). Under perspective projections the image coordinates are given as:

$$(x, y) = f\frac{(X, Y)}{Z}, \tag{3.27}$$

where f is the focal length of the sensor (camera). Sometimes, there are slightly different focal lengths in the x and y dimensions. In this case we denote these focal lengths as f_x and f_y. When we can only compute Z_x and Z_y, then using the chain rule we have:

$$\left(\frac{\partial Z}{\partial X}, \frac{\partial Z}{\partial Y}\right) = \left(\frac{\partial Z}{\partial x}\frac{\partial x}{\partial X}, \frac{\partial Z}{\partial y}\frac{\partial y}{\partial Y}\right) = \left(\frac{\partial Z}{\partial x}\frac{f}{Z}, \frac{\partial Z}{\partial y}\frac{f}{Z}\right). \tag{3.28}$$

The 3D range constraint equation then becomes:

$$Z_x U + Z_y V + \frac{Z}{f}W + \frac{Z}{f}Z_t = 0. \tag{3.29}$$

We can compute $A_{n\times 3}\vec{V} = B_{n\times 1}$, where the i^{th} row of A has entries Z_x, Z_y and $\frac{Z}{f}$ and the i^{th} row of B has entries $-\frac{Z}{f}Z_t$. The least squares solution for $\vec{V} = (U, V, W)^T$ is:

$$\vec{V} = (A^T A)^{-1} A^T B. \tag{3.30}$$

The eigenvalues $\hat{e}_0$, $\hat{e}_1$ and $\hat{e}_2$ and their corresponding eigenvalues, $\lambda_0 \leq \lambda_1 \leq \lambda_2$, can be computed from the 3×3 symmetric matrix $A^T A$ and can be used to compute a reliable least squares full range velocity, $\vec{V}$, when the smallest eigenvalue, λ_0 is greater than some threshold τ.

3.3.2 Least Squares Range and Optical Flow

It is possible to compute $(U, V, W)^T$ using both intensity and range derivatives as described in the paper "Dense Range Flow from Depth and Intensity Data" by Spies et al. 2000 [10]. We need to minimize:

$$\int\int\int \beta^2(UI_X + VI_Y + I_t) + (UZ_X + VZ_Y + W + Z_t)dXdYdWdt. \tag{3.31}$$

This yields the linear functional:

$$U(\beta^2 I_X + Z_X) + V(\beta^2 I_Y + Z_Y) + W + (\beta^2 I_t + Z_t) = 0, \tag{3.32}$$

which can be solved in local $(n \times n)$ neighborhoods by a least squares calculation.

Again, because we can only compute I_x and I_y (and not I_X and I_Y) because we are using perspective and not orthographic projection), the constraint equation becomes:

$$U(\beta^2 I_x + Z_x) + V(\beta^2 I_y + Z_y) + \frac{Z}{f}W + \frac{Z}{f}(\beta^2 I_t + Z_t) = 0 \tag{3.33}$$

β^2 is a weighting term that can take into account the relative magnitude of the intensity and range derivatives. β can be computed separately for each local neighborhood, as in general, there is not a good correlation between intensity gradient and range gradient values across the image. Typically, intensity derivatives are 2 orders of magnitude larger than depth derivatives. In that case, a value of $\beta \in [0.01, 0.1]$ might better balance the influence of the two types of derivatives. Alternatively, we could compute β as suggested by Spies et al. [10] for a $2r + 1 \times 2r + 1$ neighbourhood as:

$$\beta(x, y) = \frac{\sum_{i=x-r}^{x+r} \sum_{j=y-r}^{y+r} Z(i, j)}{\sum_{i=x-r}^{x+r} \sum_{j=y-r}^{y+r} I(i, j)}. \tag{3.34}$$

$\beta = 0$ reduces the solution to the least squares range flow solution.

As for the purely range flow case in the previous section, we can set up and solve a linear least squares system of equations, $A_{n\times 3} = vecV = B_{n\times 1}$, where the i^{th} row of A as entries $(\beta^2 I_x + Z_x)$, $(\beta^2 I_y + Z_y)$ and $\frac{Z}{f}$ and the i^{th} rows of B have entries $-\frac{Z}{f}(\beta^2 I_t + Z_t)$. The least squares solution with eigenvalue thresholding are as in the previous section.

3.3.3 Regularized Range Flow

We show how to directly compute range flow from range (depth map) derivatives as described in the paper "Regularized Range Flow" by Spies et al. 2000 [22]. We minimize:

$$\int\int\int\int F(X,Y,t,U,V,W,U_X,U_Y,U_Z,U_t,$$
$$V_X,V_Y,V_Z,V_t,W_X,W_Y,W_Z,W_t)\partial X\partial Y\partial Z\partial t, \tag{3.35}$$

where the functional to be minimized is:

$$F(X,Y,t,U,V,W,U_X,U_Y,U_t,U_Z,V_X,V_Y,V_Z,V_t,W_X,W_Y,W_Z,W_t) =$$
$$(Z_XU + Z_YV + W + Z_t)^2 +$$
$$\alpha^2(U_X^2 + U_Y^2 + U_Z^2 + U_t^2 + V_X^2 + V_Y^2 + V_Z^2 + V_t^2 +$$
$$W_X^2 + W_Y^2 + W_Z^2 + W_t^2) = 0. \tag{3.36}$$

Here we have used the fact that $Z_Z = 1$ and $Z_{ZZ} = 0$ for range data (which is a 3D surface in the depth data). Again, as in the Least Squares case, we actually can only compute Z_x and Z_y so the functional to be minimized becomes:

$$F(x,y,t,U,V,W,U_Y,U_Y,U_Z,U_t,V_X,V_Y,V_Z,V_t,W_X,W_Y,W_Z,W_t) =$$
$$(Z_xU + Z_yV + \frac{Z}{f}W + \frac{Z}{f}Z_t)^2 +$$
$$\alpha^2(U_X^2 + U_Y^2 + U_Z^2 + U_t^2 + V_X^2 + V_Y^2 + V_Z^2 + V_t^2 +$$
$$W_X^2 + W_Y^2 + W_Z^2 + W_t^2) = 0, \tag{3.37}$$

where we have again used the chain rule so that Z_x and Z_y can be used in place of Z_X and Z_Y. We minimize the general 3D Euler-Lagrange equations:

$$F_U - \frac{\partial}{\partial X}F_{U_X} - \frac{\partial}{\partial Y}F_{U_Y} - \frac{\partial}{\partial Z}F_{U_Z} - \frac{\partial}{\partial t}F_{U_t} = 0, \tag{3.38}$$

$$F_V - \frac{\partial}{\partial X}F_{V_X} - \frac{\partial}{\partial Y}F_{V_Y} - \frac{\partial}{\partial Z}F_{V_Z} - \frac{\partial}{\partial t}F_{V_t} = 0, \tag{3.39}$$

$$F_W - \frac{\partial}{\partial X}F_{W_X} - \frac{\partial}{\partial Y}F_{W_Y} - \frac{\partial}{\partial Z}F_{W_Z} - \frac{\partial}{\partial t}F_{W_t} = 0, \tag{3.40}$$

where:

$$F_U = 2Z_x(Z_xU + Z_yV + \frac{Z}{f}W + \frac{f}{f}Z_t) \tag{3.41}$$

$$F_V = 2Z_y(Z_xU + Z_yV + \frac{Z}{f}W + \frac{f}{f}Z_t) \tag{3.42}$$

$$F_W = 2\frac{Z}{f}(Z_xU + Z_yV + \frac{Z}{f}W + \frac{Z}{f}Z_t) \tag{3.43}$$

$$F_{U_X} = 2\alpha^2 U_X \tag{3.44}$$

$$F_{U_Y} = 2\alpha^2 U_Y \tag{3.45}$$

$$F_{U_Z} = 2\alpha^2 U_Z \tag{3.46}$$

$$F_{U_t} = 2\alpha^2 U_t \tag{3.47}$$

$$F_{V_X} = 2\alpha^2 V_X \tag{3.48}$$

$$F_{V_Y} = 2\alpha^2 V_Y \tag{3.49}$$

$$F_{V_Z} = 2\alpha^2 V_Z \tag{3.50}$$

$$F_{V_t} = 2\alpha^2 V_t \tag{3.51}$$

$$\tag{3.52}$$

$$F_{W_X} = 2\alpha^2 W_X \tag{3.53}$$

$$F_{W_Y} = 2\alpha^2 W_Y \tag{3.54}$$

$$F_{W_Z} = 2\alpha^2 W_Z \tag{3.55}$$

$$F_{W_t} = 2\alpha^2 W_t \tag{3.56}$$

and

$$\frac{dF_{U_X}}{dX} = 2\alpha^2 U_{XX} \tag{3.57}$$

$$\frac{dF_{U_Y}}{dY} = 2\alpha^2 U_{YY} \tag{3.58}$$

$$\frac{dF_{U_Z}}{dZ} = 2\alpha^2 U_{ZZ} \tag{3.59}$$

$$\frac{dF_{U_t}}{dt} = 2\alpha^2 U_{tt} \tag{3.60}$$

$$\frac{dF_{V_X}}{dX} = 2\alpha^2 V_{XX} \tag{3.61}$$

$$\frac{dF_{V_Y}}{dY} = 2\alpha^2 V_{YY} \tag{3.62}$$

$$\frac{dF_{V_Z}}{dZ} = 2\alpha^2 V_{ZZ} \tag{3.63}$$

$$\frac{dF_{V_t}}{dt} = 2\alpha^2 V_{tt} \tag{3.64}$$

$$\frac{dF_{W_X}}{dX} = 2\alpha^2 W_{XX} \tag{3.65}$$

$$\frac{dF_{W_Y}}{dY} = 2\alpha^2 W_{YY} \tag{3.66}$$

$$\frac{dF_{W_Z}}{dZ} = 2\alpha^2 W_{ZZ} \tag{3.67}$$

$$\frac{dF_{W_t}}{dt} = 2\alpha^2 W_{tt}. \tag{3.68}$$

Note that we have used the spatio-temporal gradients here. However, if U_t, V_t, W_t, U_{tt}, V_{tt} and W_{tt} are unknown then we just use the spatial gradient and use 0 for these terms.

Since $\nabla^2 U = U_{XX} + U_{YY} + U_{ZZ} + U_{tt}$, $\nabla^2 V = V_{XX} + V_{YY} + V_{ZZ} + V_{tt}$ and

$\nabla^2 W = W_{XX} + W_{YY} + W_{ZZ} + W_{tt}$ we can rewrite the Euler-Lagrange equations as:

$$Z_x^2 U + Z_x Z_y V + Z_x \frac{Z}{f} W + \frac{Z}{f} Z_x Z_t = \alpha^2 \nabla^2 U \tag{3.69}$$

$$Z_x Z_y U + Z_y^2 V + Z_y \frac{Z}{f} W + \frac{Z}{f} Z_y Z_t = \alpha^2 \nabla^2 V \tag{3.70}$$

$$\frac{Z}{f} Z_x U + \frac{Z}{f} Z_y V + \left(\frac{Z}{f}\right)^2 W + \left(\frac{Z}{f}\right)^2 Z_t = \alpha^2 \nabla^2 W. \tag{3.71}$$

Using the approximations $\nabla^2 U \approx \bar{U} - U$, $\nabla^2 V \approx \bar{V} - V$ and $\nabla^2 W \approx \bar{W} - W$, we can rewrite the Euler-Lagrange equations as:

$$(\alpha^2 + Z_x^2) U + Z_x Z_y V + Z_x \frac{Z}{f} W = (\alpha^2 \bar{U} - \frac{Z}{f} Z_x Z_t), \tag{3.72}$$

$$Z_x Z_y U + (\alpha^2 + Z_y^2) V + Z_y \frac{Z}{f} W = (\alpha^2 \bar{V} - \frac{Z}{f} Z_y Z_t) \text{ and} \tag{3.73}$$

$$\frac{Z}{f} Z_x U + \frac{Z}{f} Z_y V + \left(\alpha^2 + \left(\frac{Z}{f}\right)^2\right) W = (\alpha^2 \bar{W} - \left(\frac{Z}{f}\right)^2 Z_t), \tag{3.74}$$

or in matrix form as:

$$\underbrace{\begin{bmatrix} (\alpha^2 + Z_x^2) & Z_x Z_y & \frac{Z}{f} Z_x \\ Z_x Z_y & (\alpha^2 + Z_y^2) & \frac{Z}{f} Z_y \\ \frac{Z}{f} Z_x & \frac{Z}{f} Z_y & \left(\alpha^2 + \left(\frac{Z}{f}\right)^2\right) \end{bmatrix}}_{A} \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} (\alpha^2 \bar{U} - \frac{Z}{f} Z_x Z_t) \\ (\alpha^2 \bar{V} - \frac{Z}{f} Z_y Z_t) \\ (\alpha^2 \bar{W} - \left(\frac{Z}{f}\right)^2 Z_t) \end{bmatrix}. \tag{3.75}$$

Note that the 3×3 matrix on the left hand side of this equation is denoted as A. Then the Gauss Seidel iterative equations can be written as:

$$\begin{bmatrix} U^{n+1} \\ V^{n+1} \\ W^{n+1} \end{bmatrix} = A^{-1} \begin{bmatrix} (\alpha^2 \bar{U}^n - \frac{Z}{f} Z_x Z_t) \\ (\alpha^2 \bar{V}^n - \frac{Z}{f} Z_y Z_t) \\ (\alpha^2 \bar{W}^n - \left(\frac{Z}{f}\right)^2 Z_t) \end{bmatrix}. \tag{3.76}$$

3.3.4 Regularized Range Flow using Intensity and Range Derivatives

It is possible to compute $\vec{V} = (U, V, W)^T$ using both intensity and range derivatives and the same smoothing term as above. This was shown in the paper "Dense Range Flow from Depth and Intensity Data" by Spies et al. 2000 [10]. We need to minimize:

$$F = (UZ_x + VZ_y + \frac{Z}{f}W + \frac{Z}{f}Z_t)^2 + \beta^2(UI_x + VI_y + \frac{Z}{f}I_t)^2$$

$$+\alpha^2(U_X^2 + U_Y^2 + U_Z^2 + U_t^2 + V_X^2 + V_Y^2 + V_Z^2 + V_t^2 + W_X^2 + W_Y^2 + W_Z^2 + W_t^2), \quad (3.77)$$

i.e. we need to minimize $\int\int\int\int F \; dXdYdWdt$. The first three Euler Lagrange equations (see Equations (3.38) to (3.40) above) become:

$$F_U = 2Z_x(Z_xU + Z_yV + \frac{Z}{f}W + \frac{Z}{f}Z_t) + 2\beta^2 I_x(I_xU + I_yV + \frac{Z}{f}I_t), \quad (3.78)$$

$$F_V = 2Z_y(Z_xU + Z_yV + \frac{Z}{f}W + \frac{Z}{f}Z_t) + 2\beta^2 I_y(I_xU + I_yV + \frac{Z}{f}I_t) \text{ and} \quad (3.79)$$

$$F_W = 2\left(\frac{Z}{f}Z_xU + \frac{Z}{f}Z_yV + \left(\frac{Z}{f}\right)^2 W + \left(\frac{Z}{f}\right)^2 Z_t\right). \quad (3.80)$$

The other derivatives are the same as in Equations (3.44) to (3.68) Since $\nabla^2 U = U_{XX} + U_{YY} + U_{ZZ} + U_{tt}$, $\nabla^2 V = V_{XX} + V_{YY} + V_{ZZ} + V_{tt}$ and $\nabla^2 W = W_{XX} + W_{YY} + W_{ZZ} + W_{tt}$ we can rewrite the Euler-Lagrange equations as:

$$Z_x(Z_xU + Z_yV + \frac{Z}{f}W + \frac{Z}{f}Z_t) + \beta^2 I_x(I_xU + I_yV + \frac{Z}{f}I_t) = \alpha^2\nabla^2 U \quad (3.81)$$

$$Z_y(Z_xU + Z_yV + \frac{Z}{f}W + \frac{Z}{f}Z_t) + \beta^2 I_y(I_xU + I_yV + \frac{Z}{f}I_t) = \alpha^2\nabla^2 V \quad (3.82)$$

$$\frac{Z}{f}Z_xU + \frac{Z}{f}Z_yV + \left(\frac{Z}{f}\right)^2 W + \left(\frac{Z}{f}\right)^2 Z_t = \alpha^2\nabla^2 W. \quad (3.83)$$

Using the approximations $\nabla^2 U \approx \bar{U} - U$, $\nabla^2 V \approx \bar{V} - V$ and $\nabla^2 W \approx \bar{W} - W$ as above, we can rewrite the Euler-Lagrange equations as:

$$\begin{aligned}(\alpha^2 + Z_x^2 + \beta^2 I_x^2)U + (Z_x Z_y + \beta^2 I_x I_y)V + Z_x \frac{Z}{f} W &= \\ (\alpha^2 \bar{U} - \frac{Z}{f}(Z_x Z_t + \beta^2 I_x I_t) & \qquad (3.84)\\ (Z_x Z_y + \beta^2 I_y^2)U + (\alpha^2 + Z_y^2 + \beta^2 I_y^2)V + Z_y \frac{Z}{f} W &= \\ (\alpha^2 \bar{V} - \frac{Z}{f}(Z_y Z_t + \beta^2 I_y I_t)) & \qquad (3.85)\\ \frac{Z}{f} Z_x U + \frac{Z}{f} Z_y V + \left(\alpha^2 + \left(\frac{Z}{f}\right)^2\right) W &= \\ (\alpha^2 \bar{W} - \left(\frac{Z}{f}\right)^2 Z_t). & \qquad (3.86)\end{aligned}$$

In matrix form, this becomes:

$$\underbrace{\begin{bmatrix} (\alpha^2 + Z_x^2 + \beta^2 I_x^2) & (Z_x Z_y + \beta^2 I_x I_y) & \frac{Z}{f} Z_x \\ (Z_x Z_y + \beta^2 I_x I_y) & (\alpha^2 + Z_y^2 + \beta^2 I_y^2) & \frac{Z}{f} Z_y \\ \frac{Z}{f} Z_x & \frac{Z}{f} Z_y & \left(\alpha^2 + \left(\frac{Z}{f}\right)^2\right) \end{bmatrix}}_{=A} \begin{bmatrix} U \\ V \\ W \end{bmatrix} =$$

$$\begin{bmatrix} \alpha^2 \bar{U} - \frac{Z}{f}(Z_x Z_t + \beta^2 I_x I_t) \\ \alpha^2 \bar{V} - \frac{Z}{f}(Z_y Z_t + \beta^2 I_y I_t) \\ \alpha^2 \bar{W} - \left(\frac{Z}{f}\right)^2 Z_t \end{bmatrix}. \qquad (3.87)$$

When $\beta = 0$ we have standard range flow regularization as given by Equation (3.76). Thus, the Gauss Seidel iterative equations for range/intensity flow can be written as:

$$\begin{bmatrix} U^{n+1} \\ V^{n+1} \\ W^{n+1} \end{bmatrix} = A^{-1} \begin{bmatrix} (\alpha^2 \bar{U}^n - \frac{Z}{f}(Z_x Z_t + \beta^2 I_x I_t)) \\ (\alpha^2 \bar{V}^n - \frac{Z}{f}(Z_y Z_t + \beta^2 I_y I_t)) \\ (\alpha^2 \bar{W}^n - \left(\frac{Z}{f}\right)^2 Z_t) \end{bmatrix} \qquad (3.88)$$

3.4 Mathematical Equivalence of Scene Flow and Range Flow

Since the disparity d is the perspective projection of the baseline b, scene flow $(X', Y', Z')^T$ and range flow $(U, V, W)^T$ are the same thing theoretically and this can be expressed mathematically by the following equation shown below.

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} \frac{\partial X}{\partial t} \\ \frac{\partial Y}{\partial t} \\ \frac{\partial Z}{\partial t} \end{pmatrix} = \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} X_t - X_{t-1} \\ Y_t - Y_{t-1} \\ Z_t - Z_{t-1} \end{pmatrix} = b \begin{pmatrix} \frac{x-x_0}{d} - \frac{x+u-x_0}{d+p} \\ \frac{y-y_0}{d} - \frac{y+v-y_0}{d+p} \\ \frac{f_x}{d} - \frac{f_x}{d+p}, \end{pmatrix} \tag{3.89}$$

where f_x is the focal length in the x direction. Thus, given u, v, d, p we compute U, V, W.

3.5 Inversely solving for u, v, p from U, V, W

We can also go the other way: given U, V, W we can compute u, v, p. Note that if we have computed U, V, W from range data then u, v, p are for the equivalent scene (imaginary) data.

Given $W(x, y)$, we can solve for $p(x, y)$ as:

$$p(x, y) = -\frac{d^2(x, y) * W(x, y)}{d(x, y) * W(x, y) + B * f_x}. \tag{3.90}$$

Note that here we have used f_x (rather than f) as the focal length in case $f_x \neq f_y$. Then given $U(x, y)$, we can solve for $u(x, y)$ as:

$$u(x, y) = \frac{(d(x, y) * p(x, y) + d^2(x, y)) * U(x, y) + (p(x, y) * x - p(x, y) * x_0) * B}{d(x, y) * B} \tag{3.91}$$

and finally given $V(x,y)$, we can solve for $v(x,y)$ as:

$$v(x,y) = \frac{(d(x,y) * p(x,y) + d^2(x,y)) * V(x,y) + (p(x,y) * y - p(x,y) * y_0) * B}{d(x,y) * B}. \tag{3.92}$$

Note that while U, V, W are theoretically constant almost everywhere (except for non-stationary objects in the sequence) we use x, y to denote that in practice they can be different (due to measurement error, etc.).

Then we can warp the left images with $(u_L, v_L) = (u, v)$ and the right images with $(u_R, v_R) = (u + p, v)$ to remove this motion. Suppose L_1, L_2 and R_1, R_2 are the two adjacent left and right images respectively, at two time instances. Then, we could warp L_2 into L_1 using (u_L, v_L) and could warp R_2 into R_1 using (u_R, v_R). Of course, for range flow we do not explicitly use the right images. The Z values (or disparity values d could come from stereo image sequences as we do here but they could be measured directly by a range sensor or computed indirectly from a monocular motion and structure calculation. We explain monocular motion and structure calculation as follows. A motion and structure algorithm computes the camera motion and depth map of the environment from time varying optical flow. Since only a a single camera is used, the quantities are relative and not absolute. Thus, if one object is 10m away and another is 5m away then we can only say that the first object is twice as far away as the second object but we cannot give any absolute depth information. The motion parameters describe the camera translation and rotation. Rotation is independent of depth and so can be fully recovered. On the other hand, only the direction of translation (scaled by a depth factor) can be recovered.

3.6 Warping Range Flow

We can also warp the depth map $Z2$ into $Z1$ using dW, which is the incremental change in W by subtracting dW from $Z2$ via simple difference. $Z2$ approaches closer

to $Z1$ as a result of this.

3.7 Basic Similarities and Differences Between Scene Flow and Range Flow Methods

Both scene flow and range flow are the 3D optical flow on visible environmental object surfaces. For a stationary scene and moving observer, all these vectors should be in the same direction and have the same magnitude. For non-stationary scenes, independently moving objects should have different 3D vectors, whose directions and magnitudes correspond to their motion relative to the observer. Methods for computing scene flow involve using depth information in conjunction with left and right 2D optical flows computed for a stereo image sequence to compute 3D optical flow. As such, it is both depth and intensity based. Range flow, on the other hand, involves using a range flow constraint equation on the depth values and their 1^{st} order derivatives to recover 3D optical flow. Sometimes, range flow is depth based only. However, range flow can also be depth and intensity based when intensity information of the images is incorporated in the range flow algorithm. Another minor difference between range flow and scene flow is that the scene flow algorithm takes the disparity d as input while the range flow takes the depth map Z as input (of course, $Z = \frac{fB}{d}$, where B is the stereo baseline and f is the focal length).

Chapter 4

Experimental Technique

This chapter describes the tools used to implement and evaluate our two algorithms. We begin this chapter by describing the datasets we used when testing these algorithms. We give the extrinsic and intrinsic parameters of the camera for these datasets. Then we present the filters we used to compute the 1^{st} order derivatives of the images and the depth maps. We present descriptions of the two stereo algorithms we used to calculate the disparities and the depth maps. We also explain other techniques we used to implement the algorithms such as the need to use bilinear sampling. Afterward, we explain the methodology used to display the 3D scene and range flow fields. Finally, we close this chapter with a description of our programming environment.

4.1 Experimental Datasets

This section describes the synthetic and real datasets that we used in our analysis. Both the real and the synthetic datasets are stereo image sequences of traffic environments. Images of these traffic sequence are shown in Figure 4.1. The image data used in this thesis is publicly available at

http://www.mi.auckland.ac.nz/EISATS

4.1.1 Synthetic Dataset

The synthetic data is a driver assistance stereo image sequence provided by Toby Vaudrey (University of Auckland) and Clemens Rabe (Daimler AG) from the website

http://www.mi.auckland.ac.nz/
index.php?option=com_content&view=article&id=162&Itemid=67

The website refers to these images as Sequence 1 and Sequence 2. We used both synthetic Sequence 1 and Sequence 2 to do our experiments. The ground truth for stereo disparity (d), stereo disparity gradient (p) 2D left image optical flow (u, v) and the extrinsic and intrinsic camera parameters were also provided for both of these sequences. Sequence 1 is a stereo sequence is 100 image pairs long, while Sequence 2 has 396 frames, and both were taken with perfectly calibrated stereo cameras as mentioned in Vaudrey et al. [25]. The resolutions of images in both the stereo sequences were 640×480 pixels.

4.1.2 Real Data Set

The real dataset we used is called the "construction site" sequence. Daimler AG, which is a German car manufacturer, provided Wedel et al. [29] with this traffic sequence for research purposes on the website

http://www.mi.auckland.ac.nz/
index.php?option=com_content&view=article&id=45&Itemid=67\$

The "construction site" sequence features normal traffic density on an expressway, with a standard safety fence between opposite traffic directions. However, normal lanes have been cut in width and shifted to the right, with several slow large trucks in the right lane. This sequence has been captured with a calibrated stereo pair of 12-bit grey-scale cameras near Stuttgart, Germany. This sequence contains 300 pairs of frames. The intrinsic and extrinsic camera pairs were also made available. The resolution of the images is 640 × 481 pixels and they are saved in PGM (Portable Grey Map) format.

(a) Real data at frame $t = 11$ (b) Synthetic data at frame $t = 11$ (c) Synthetic data at frame $t = 215$

Figure 4.1: Synthetic and Real Images used in the experiment: (a) The real data at frame $t = 11$, (b) Synthetic data at frame $t = 11$ for Sequence 1 and (c) Synthetic data at frame $t = 215$ for Sequence 2.

4.2 Camera Parameters for Synthetic Data

We show the extrinsic and intrinsic camera parameters for the synthetic data as provided by a website

```
http://www.mi.auckland.ac.nz/DATA/6D/Set02/camera.dat
```

```
[INTERNAL]
F =1   # [m] focal length
SX=0.001225   # [m] pixel size in X direction
```

```
SY=0.00122   # [m] pixel size in Y direction
X0=319.5   # [pixel] X-coordinate of principle from top-left (0,0)
Y0=239.5   # [pixel] X-coordinate of principle from top-left (0,0)


[EXTERNAL]
B =0.3   # [m] width of baseline of stereo camera rig
LATPOS=0   # [m] lateral position of rectified images (virtual camera)
HEIGHT=0   # [m] height of rectified images (virtual camera)
DISTANCE=0   # [m] distance of rectified images (virtual camera)
TILT =0   # [rad] tilt angle
YAW=0   # [rad] yaw angle
ROLL=0   # [rad] roll angle

# Notes:
#
#  In a stereo camera system the internal parameters
#  for both cameras are the same.
#
#  The camera model is right handed.
#    The X axis is the lateral distance (positive to the right)
#    The Y axis is the height (positive pointing up)
#    The Z axis is in the depth direction (positive in the
#    direction of driving)
#
#  The world to camera transformation is performed by first a translation
#  (latpos, height, distance) followed by a rotation (tilt, yaw, roll).
#
#  The angle directions are:
#   tilt > 0  <=>  looking down
```

```
#   yaw  > 0  <=>  looking right
#   roll > 0  <=>  rolling clockwise
#
```

4.3 Camera Parameters for Real Data

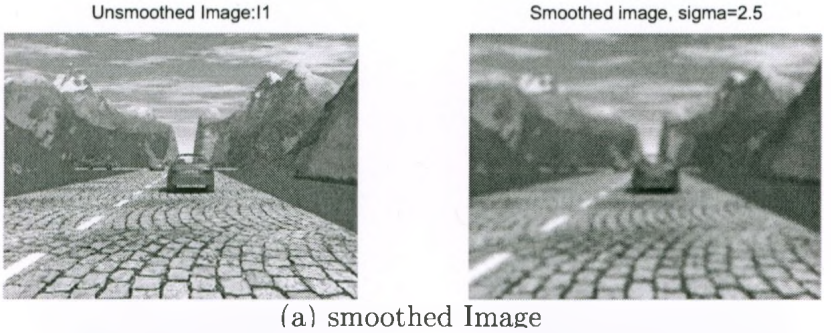
The camera parameters and their descriptions are shown below. We obtained these values from technical report CITR-TR-207, Computer Science, University of Auckland, " Performance Evaluation of Stereo and Motion i Analysis on Rectified Image Sequences" by Zhifeng Liu and Reinhard Klette [16].

```
############################################################
#   Camera parameter file for ts_StereoCamera class.              #
############################################################

[INTERNAL]
F  = 820.428  # [pixel] focal length
SX = 1.0      # [pixel] pixel size in X direction
SY = 1.000283 # [pixel] pixel size in Y direction
X0 = 305.278  # [pixel] X-coordinate of principle
Y0 = 239.826  # [pixel] Y-coordinate of principle

[EXTERNAL]
B  =  0.308084  # [m] width of baseline of stereo camera rig
LATPOS = -0.07  # [m] lateral position of rectified images (virtual camera)
HEIGHT =  1.26  # [m] height of rectified images (virtual camera)
DISTANCE =  2.0  # [m] distance of rectified images (virtual camera)
TILT     =  0.06 # [rad] tilt angle
YAW      = -0.01 # [rad] yaw angle
ROLL     =  0.0  # [rad] roll angle
```

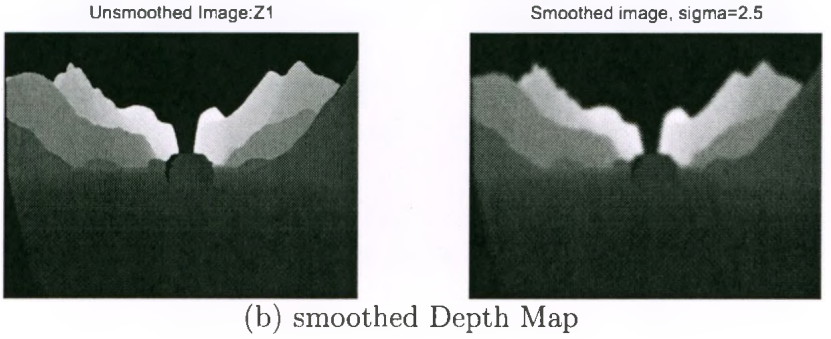
Note that the TILT, YAW and ROLL angles are essentially zero, meaning the 3D motion is pure translation. TILT is the pitch angle, i.e. rotation about the x axis. YAW is rotation about the y axis while ROLL is rotation about the longitudinal axis.

4.4 Gaussian Smoothing

There is a lot of noise in the depth maps as well as the images. We applied 2D separable Gaussian filtering with $\sigma = 2.5$ to blur the images and reduce noise effects. We show the result of Gaussian blurring for one image and its depth map in Figure 4.2.



(a) smoothed Image



(b) smoothed Depth Map

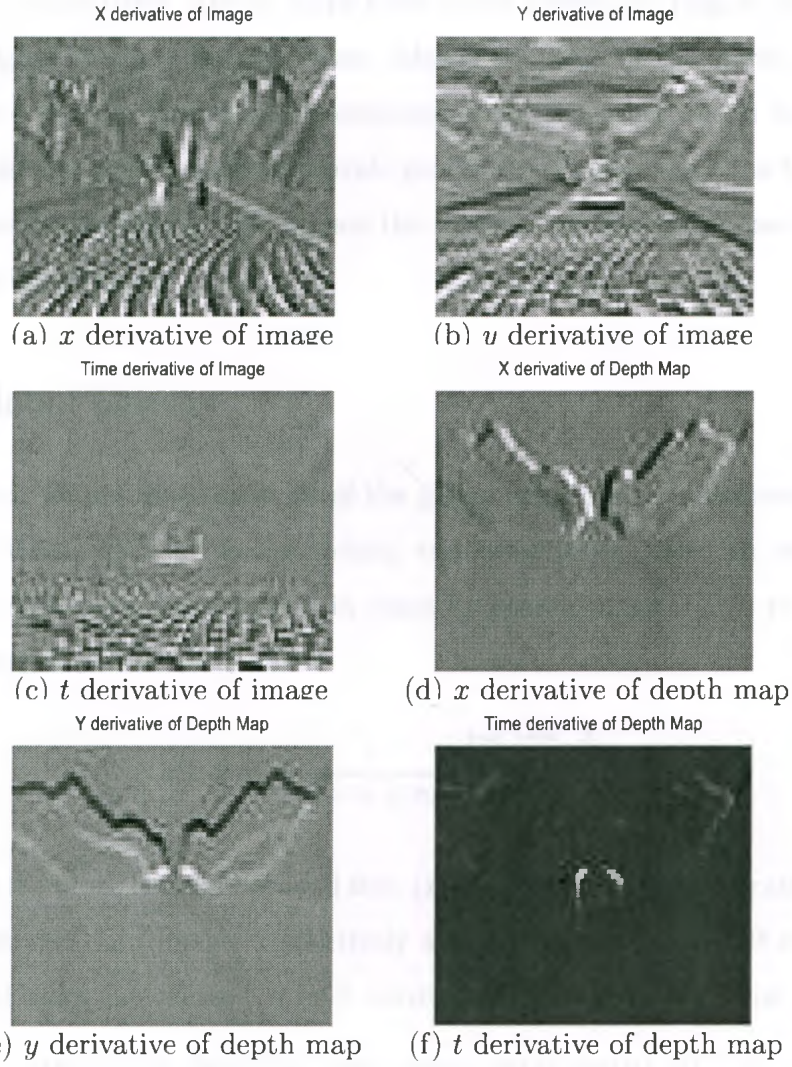
Figure 4.2: The 11^{th} *image and depth map before and after Gaussian smoothing with* $\sigma = 2.5$.

4.5 Differentiation using Filters

To compute 1^{st} order derivatives of the blurred images and their depth maps, we used two 4 point central difference filtering. Thus the 5 tap filter is $[-1\ 8\ 0\ -8\ 1]/12$. We used 4 point central difference as this filter is used by Brox et al. [6]. The numerical calculations using 4 point central difference are better than those using 2 point central difference. Other 5 tap filters (the number of coefficients), such as Simoncelli [21] have similar performance to 4 point central difference.

In our range flow algorithm and the implementation of Wedel et al.'s scene flow algorithm, we used 4 point central difference to compute the x and y spatial derivatives. We used simple pixel differences to compute the temporal derivatives because Wedel et al. [28, 29] also used simple difference to compute these temporal derivatives.

We show some derivative images of intensity and depth maps in Figure 4.3.



(a) x derivative of image (b) u derivative of image

(c) t derivative of image (d) x derivative of depth map

(e) y derivative of depth map (f) t derivative of depth map

Figure 4.3: Some derivative images: (a)-(c) the spatio-temporal intensity derivative images and (d)-(f) the spatial-temporal depth derivative images for the 11^{th} *synthetic image of sequence 1.*

4.6 Stereo Algorithms

To compute the disparity d and the depth maps Z, we used the two stereo algorithms provided in a paper titled "Depth Maps From Color Images By Region Based Stereo Matching Algorithms" by Baris Baykant Alagoz [1]. One of the stereo algorithms is global error energy minimization technique while the other one is line growing method. These algorithms seem to provide good disparity maps and the MatLab implementations were available. We explain the concepts of these two stereo algorithms in the upcoming sections.

4.6.1 Global Energy

In this section, we describe the steps of the global error energy minimization stereo algorithm. Alagoz used a block-matching technique to calculate an error energy matrix $e(i, j, d)$ for every disparity d in disparity search range (i.e. 0 to maximum disparity) using Equation (4.1) below.

$$e(i,j,d) = \frac{1}{3 \times n \times m} \cdot \sum_{x=i}^{i+n} \sum_{y=j}^{j+m} \sum_{k=1}^{3} \tag{4.1}$$

Here $n \times m$ is the block or neighborhood size, (x, y) and (i, j) are pixel locations, L and R are the left and right images respectively and k represents the RGB components of images and takes the values 1, 2 or 3, corresponding to red , blue, and green.

In the next step of the algorithm, every error energy matrix $e(i, j, d)$ is smoothed by applying an averaging filter. The averaging filter removes very sharp changes in energy, which may be caused by incorrect matching. The averaged values of $e(i, j, d)$ are represented by $\bar{e}(i, j, d)$ in Equation (4.2):

$$\bar{e}(i,j,d) = \frac{1}{n \times m} \sum_{x=i}^{i+n} \sum_{y=j}^{j+m} e(x,y,d). \tag{4.2}$$

Finally, for every (i, j) pixel location, Alagoz selected the d value that had minimum error energy $\bar{e}(i, j, d)$, as the most reliable disparity estimation for pixel (i, j) and assigned it to $d(i, j)$.

4.6.2 Region Growing

The line growing stereo algorithm uses the following three steps to compute the disparity and the depth map. In step 1, since disparity of stereo images only exist in row or x directions due to the epipolar constraint, we search along the rows to find a root point, which does not belong to any grown region and then find its disparity using the energy function in Equation (4.3):

$$e(i, j, d) = \frac{1}{3 \times n \times m} \times \sum_{x=i}^{i+n} \sum_{y=j}^{j+m} \sum_{k=1}^{3} \tag{4.3}$$

If the error energy of a selected point is equal to or lower than a line growing threshold, we select it as root point disparity and go to step 2. If no disparity with lower enough energy was found, we repeat step 1 for the next point in the row.

In step 2 of the algorithm, Alagoz calculates the error energy of the neighbour points adjacent to the root point disparity. If the error energy is less than or equal to some predetermined error energy threshold (line growing threshold), we associate this point with the region. Otherwise, we return to step 1 to find a new root point. Note that a point being associated with a root point means that point has the same disparity as the root point. Thus, the pixels in a region formed from all associate points have the same disparity value.

Finally, in step 3 of the algorithm, Alagoz performs Step 1 and Step 2, row by row, until the last point in the image is reached. Thus, the grown disparity regions are composed of the disparity map $d(i, j)$.

4.6.3 Converting Pixels to Meters

We used the "construction site" sequence provided by Daimler AG as the real dataset. Daimler gave the focal length in pixels (820.428) and the pixel sizes in pixels (1.0 and 1.000283). We used the stereo matching algorithms provided by Alagoz [1]. These algorithms require that the baseline and the focal length be in meters. Therefore, we had to convert the focal length and baseline to meters for the real data. In order to do the conversion, we asked Reinhard Klette at the University of Auckland about the details of the Bosch camera, which was used to image the traffic sequence. The formula to convert focal length in pixels to meters is:

```
F[px]*pixelSixe=F[m],
```

where F is the focal length. The details of the left camera are given below.

```
#  #####  Details of left recording camera ##########
# Pixel size   = 8 micrometers (0.000008 [m])
# Sensor size  = 5.12 x 3.84 millimeters (0.00512 x 0.00384 [m])
# Focal length = 6.6563424 millimeters (0.006 [m])
# Focal length = 820.428 [px]
```

Therefore, the focal length can be converted into meters using the pixel size of the recording cameras (8 micrometers) to be 0.00656 meters for 820.428 pixels.

4.6.4 Depth Map Generation from Disparity Maps

To derive the depth map from the disparity map, Wedel et al [29, 28] used the formula given in Equation (4.4) below:

$$Z(i,j) = f \times \frac{b}{d(i,j)} \tag{4.4}$$

Here f is the focal length, d is the disparity and b is the baseline (the distance between the two stereo cameras).

4.6.5 Selection Rational for the Stereo Algorithms

As we mentioned earlier, we used these two stereo algorithms because they were both implemented in Matlab. Both algorithms used median filtering to eliminate unreliable disparities (occlusions). Occlusion occurs when there are regions in one image that the matcher cannot find a good match for in the other image of the stereo pair (probably the region is "hidden" in the second image). Both of these algorithms used median filtering to make the depth maps and the disparity maps smoother which reduced noise. Also, the line growing algorithm was more time efficient than the global error energy minimization technique. The global error energy minimization produced more reliable disparity maps, but this algorithm was more time consuming.

4.7 Converting the Gray Value Images to Color

The stereo algorithms in the paper by Baris Baykant Alagoz [1] used RGB color image for the stereo pair input. However, the Daimler data set was gray value images. Therefore, we converted the data set to color images using the Matlab 'cat' operator. The idea of this conversion is to take the grayvalue image and make it the red, green and blue images. The resulting color image will still look gray because each corresponding color tuple has the same values for red, green and blue.

4.8 Warping

Below we describe the warping performed for scene flow and range flow.

4.8.1 Pyramid Technique for Coarse to Fine Warping for Wedel et al.'s Scene Flow

A multiscale (hierarchical) approach is combined with fixed point iterations via a downsampling strategy. We downsample the full size images to build a pyramid with smaller images in each level. k is used as an index to the pyramid level. The top level number of pyramid is 4 and the re-scale factor is $\eta = 0.5$. The pyramid can go all the way to the top having the smallest possible sized images on the top level. At the top of the pyramid, the size of the image is the smallest and then both the image size and the magnitude of the scene flow get larger and larger as it goes down the pyramid. We build 4 separate pyramids for the left image at time t and $t+1$, and the right image at time t and $t+1$ respectively. Warping in such a manner reduces problems caused by aliasing.

4.8.2 Range Flow Warping

We can also warp the depth map $Z2$ into $Z1$ using dW, which is the incremental change in W by subtracting dW from $Z2$ via simple difference. $Z2$ approaches closer to $Z1$ as a result of this.

4.9 Bilinear Sampling

The scene flow algorithm in Wedel et al. 2010 [28] calculates u and v and p. We set u, v, and p initially to zeros as suggested in Wedel et al. 2010 [28]. To compute the derivatives of the left image at time $t+1$, we had to compute $\nabla L(x+u, y+v, t+1)$, where x, y are pixel locations or image coordinates. Moreover, we use 4 point central difference to calculate ∇. The variables u, v, and p are updated during the algorithm.

However, one thing to note is that $x+u$ and $y+v$ are floating point numbers.

We do not know what these values are when we only have integer values for pixels. To get values at floating point numbers, they used bilinear interpolation. Suppose $x = y = 100$ and $u = 0.236$ and $v = 0.49345$. That means we need a value at (x, y)=(100.236, 100.49345). But how would we compute this? We would simply use values at integer locations (x, y), $(x + 1, y)$, $(x, y + 1)$, $(x + 1, y + 1)$ or (100,100), (101,100), (100,101) and (101,101) respectively and appropriately weighted using bilinear weights. Matlab function `interp2` does this for us (actually this function also allows bicubic interpolation which is slightly better but computationally more expensive than bilinear interpolation).

4.10 Displaying the 3D Range Flow and 3D Scene Flow

We plotted (U, V, W) which is the range flow and scene flow against (x, y, Z) instead of plotting the range flow and scene flow against (X, Y, Z). (x, y) are the image coordinates of where the world points X, Y, Z would project to. If we plotted (U, V, W) at (X, Y, Z), we would get flow at a cluster of these values and this doesn't look qualitatively good. Using Z with x and y shows the depth at the image coordinates. How we get a dense uniformly sampled flow field. The Z values add depth to the flow and makes it 3D.

4.11 Programming Environment

We implemented both of the algorithms in MATLAB (a programming language developed by the MathWorks). MATLAB (**Mat**rix **Lab**oratory) is a tool that allows the user access to high performance numerical computation and visualization capabilities. With more than 50 sets of built-in functions, available via toolboxes, Matlab contains

many collections of functions (toolboxes) written for specialized applications, such as image processing and graphics. We use the IPT (Image Processing Toolbox) for the work described here. MATLAB also provides an interface for other programming languages such as C/C++, JAVA and Fortran. Because Matlab can call functions and subroutines written in languages such as C and Fortan, this allows users to integrate functions written in these languages into their MATLAB program. Another thing we should mention is that matrices (vectors are special cases of matrices and a scalar is an array of size 1×1) are at the heart of MATLAB, since all data in MATLAB are stored in matrices [11]. MATLAB is optimized for array operations, i.e. allows vectorized code to be written. According to results in Gonzalez, Woods and Eddins [9], for a simple application that computes sin of each element of a certain matrix, the un-vectorized function is 34 times slower than the vectorized code that computes the same result. Vectorized MATLAB code (although interpretated) is as fast as compile C code [11]. Therefore, in our programming, we vectorized our code as much as possible to get the highest efficiency.

Chapter 5

Experimental Results and Analysis

We explain the difference measures, such as Fleet's angular error, used in the quantitative evaluation of these algorithms. We present and analyze the flow fields obtained by the scene flow and range flow algorithm. We also show qualitative results on synthetic and real data.

5.1 Synthetic Data Results

We used the synthetic data, which is a driver assistance stereo image sequence provided by Toby Vaudrey (University of Auckland) and Clemens Rabe (Daimler AG) from the website

`http://www.mi.auckland.ac.nz/`
`index.php?option=com_content&view=article&id=162&Itemid=67`

for our quantitative error analysis. We used the 11^{th} image of Sequence 1 because Wedel et al. [28] used the same image to do their error analysis. They also used the 25^{th} image, but we found that the 25^{th} image gave comparable results so we just decided to show the results for the 11^{th} image. We also used the 215^{th} frame of

Sequence 2 to evaluate our range flow algorithm because it involved 2 non-stationary cars. Note that when we talk about using the 11^{th} image throughout the thesis, we mean images 11 and 12 in Sequence 1, because these are 2-frame algorithms. Similarily, the 215^{th} means images 215 and 216 in Sequence 2.

Note that these images were generated by Persistence of Vision Raytracer or povray, which is a ray tracing program. Ray tracing is a computer graphics method to generate synthetic images with moving camera and independently moving objects. Basically, a ray is cast from the camera through each image pixel out into the 3D world. What object the ray first encounters is the object seen at that pixel. A lighting calculation is done for that 3D point and the pixel is coloured that colour. More details about ray tracing can be found on the web (for example, see http://en.wikipedia.org/wiki/Ray_tracing).

These images were pgm images, which are 16bit grayvalues so we needed to use the Matlab function 'im2uint8' to rescale the data so that derivatives can be correctly computed. These images were generated by the povray ray tracing program so the intensity resolution is a program design decision.

5.1.1 Correct Range/Scene Flow

We were given the ground truth u, v, p, d values, and from them, we computed correct U, V, W values using the equations shown below in Equation (5.1) [28].

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} X_t - X_{t-1} \\ Y_t - Y_{t-1} \\ Z_t - Z_{t-1} \end{pmatrix} = b \begin{pmatrix} \frac{x+u-x_0}{d+p} - \frac{x-x_0}{d} \\ \frac{y+v-y_0}{d+p} - \frac{y-y_0}{d} \\ \frac{f_x}{d+p} - \frac{f_x}{d}, \end{pmatrix} \tag{5.1}$$

where f_x is the focal length in the x direction. However, the ground truth data had many errors, so simply plugging them into the Equation (5.1) did not give us good 3D correct scene flow. Figure 5.1 shows the 3D flow field directly computed using

Equation (5.1) without any pre-processing (occlusion disparities are correctly handled but Z values are used as given).

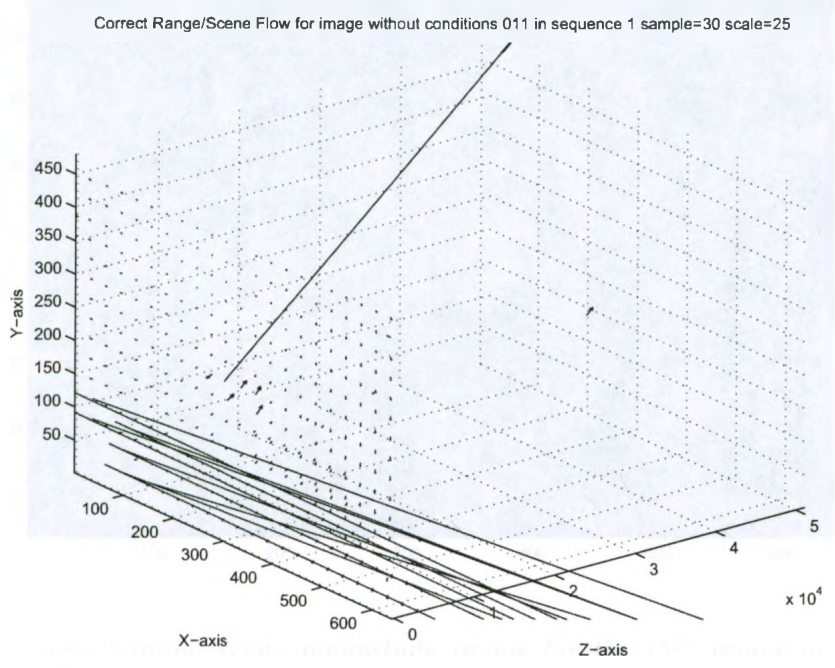


Figure 5.1: Correct range/scene flow for the 11^{th} *image of Sequence* 1 *before thresholding*

We see from Figure 5.1 that there are outliers in the data. Figure 5.2 shows the vector field in Figure 5.1 as a colour image. Note that most of the image is blue corresponding to the small magnitude velocities. The lower left part of the image corresponds to the large velocities.

For this data, the U, V, W vectors should be constant everywhere, except when there are independently moving objects (cars). Notice that in Figure 5.1, there seems to be a "wall" of tiny velocities, these are correct velocities at relatively small depth values. Removal of the Z outlier shows that these velocities are actually quite significant.

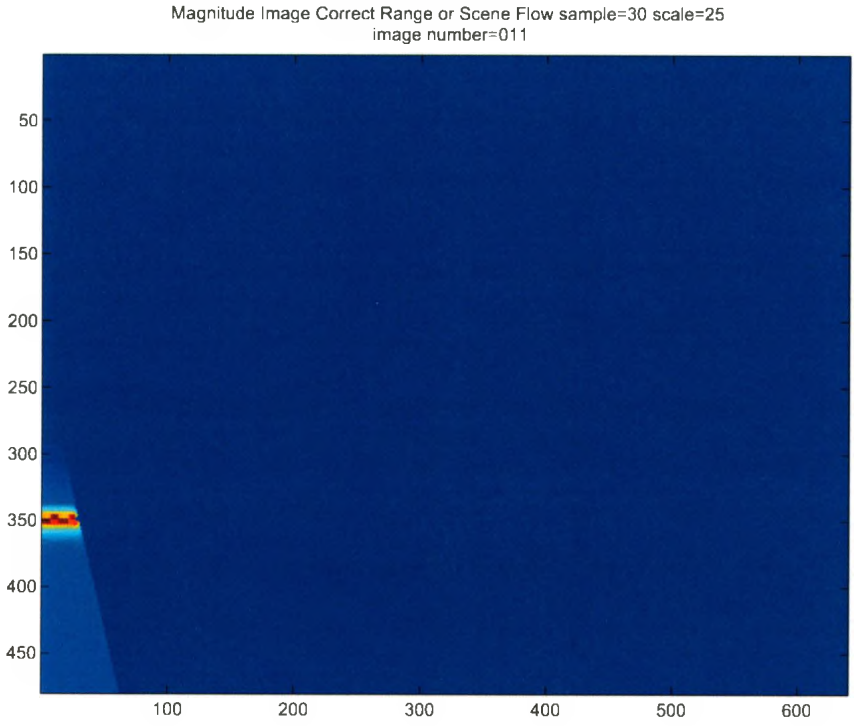


Figure 5.2: Correct range/scene magnitude image for the 11^{th} *image of Sequence* 1 *before thresholding*

5.1.2 Pre-processing Steps to Obtain Correct Range/Scene Flow

We used several pre-processing steps to obtain the correct looking flow field in Figure 5.3 and its colour magnitude image in Figure 5.4. Note that the colour image now clearly shows the three cars (red, orange and blue) and the moutain outline correctly. First, we computed the "correct" depth map using the equation $Z = *B*F./(d*SX)$, where F is the focal length, B is the baseline, d is the disparity, SX is the pixel size in X direction and ./ is vectorized division. We used histogram equalization to view the depth maps. Histogram equalization equalizes the contrast distribution in the image. Because there were many outliers in the depth maps, we thresholded those values whenever disparity was very small ($Z > 1000$). To obtain U, V, W, we made

a slight modification to Equation (5.1), as shown in Equation (5.2) below, where we have multiplied the denominator of Z_t and Z_{t-1} by SX or the pixel size.

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} X_t - X_{t-1} \\ Y_t - Y_{t-1} \\ Z_t - Z_{t-1} \end{pmatrix} = b \begin{pmatrix} \frac{x+u-x_0}{d+p} - \frac{x-x_0}{d} \\ \frac{y+v-y_0}{d+p} - \frac{y-y_0}{d} \\ \frac{F}{(d+p)*SX} - \frac{f_x}{d*SX} \end{pmatrix}. \tag{5.2}$$

We applied 7×7 median filtering to remove further local outliers and then assigned the final answers to U, V, W. We show the correct range/scene flow below in Figure 5.3 for the 11^{th} image for synthetic Sequence 1. We use the 11^{th} image because Wedel et al. [28] used that in their paper. We believe that Figure 5.3 is closer to the correct 3D display because there is significant flow in the lower part of the image where there is disparity. We also see lots of empty spaces in the flow field. This is because thresholding removed a lot of velocities. Because of occlusion, only 3.46% of the velocities were eliminated. However, the Z thresholding eliminated 22.25% of the velocities, but we can see from Figure 5.3 that still some outliers remain in the correct range/scene flow.

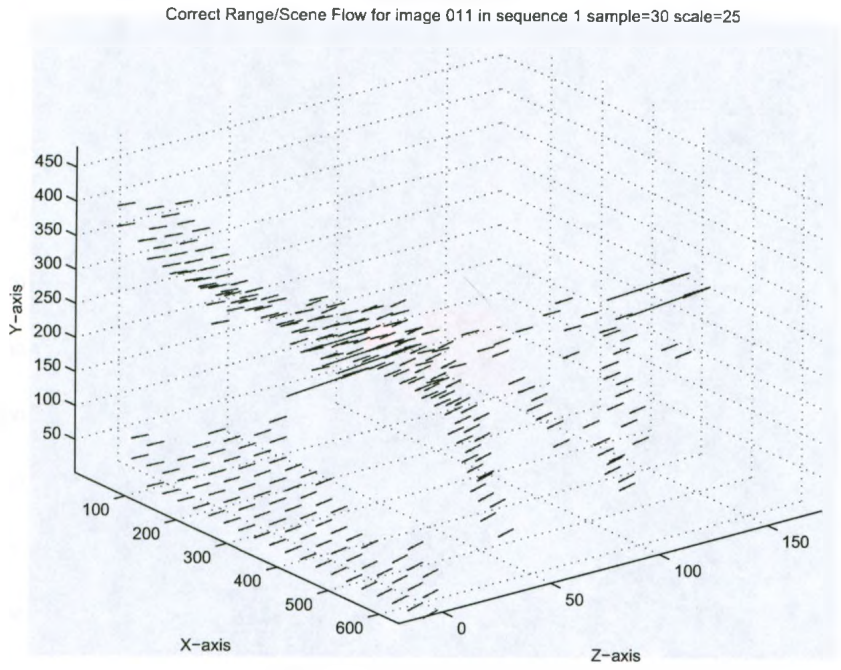


Figure 5.3: Correct range/scene flow for the 11^{th} *image of Sequence* 1 *after thresholding*

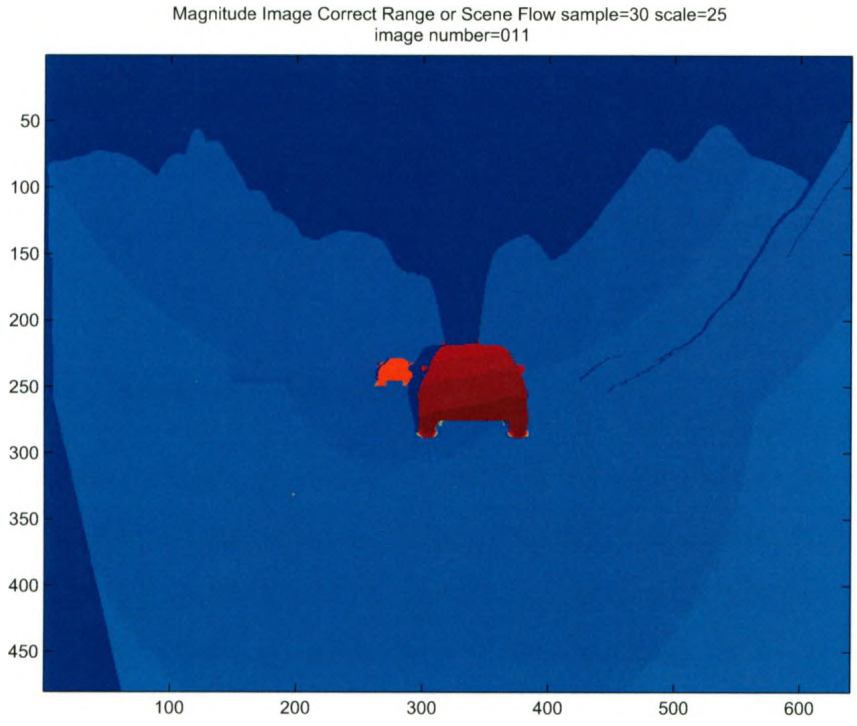


Figure 5.4: Correct range/scene magnitude image for the 11^{th} *image of Sequence* 1 *after threshol ding*

5.1.3 Measuring Flow Difference

Because we are not 100% confident that the correct flow fields are actually the correct flow fields, we are actually computing differences rather than errors. We use average angular error (or average angular difference) to do quantitative analysis on our results.

5.1.3.1 Error Analysis for Depth Maps

We use relative magnitude error to measure the error/difference in depth map values.

We also computed the differences between the "correct" depth map and the computed depth maps as given by the region growing and global energy stereo algorithms of Alagoz 2008 [1].

5.1.3.2 Error Analysis for 3D Range Flow

We used Fleet 3D average angular error to measure the difference between the correct 3D scene flow field and the computed 3D flow field from the range flow and scene flow algorithms. 3D velocity can be written as displacement per time unit as in $\tilde{\mathbf{v}} = (U, V, W)$ pixels/frame or as a space-time direction vector $(U, V, W, 1)$ in units of (pixel, pixel, pixel, frame). Let $\tilde{\mathbf{v}}_\mathbf{c} = (U_c, V_c, W_c, 1)$ represent the correct velocity at the pixel (i, j) and $\tilde{\mathbf{v}}_\mathbf{e} = (U_e, V_e, W_e, 1)$ represent the computed velocity at point (i, j, k). We can compute a normalized velocity using:

$$\hat{v} \equiv \frac{1}{\sqrt{U^2 + V^2 + W^2 + 1}}(U, V, W, 1)^T. \tag{5.3}$$

Let $\hat{\mathbf{v}}_\mathbf{c}$ and $\hat{\mathbf{v}}_\mathbf{e}$ be normalized versions of $\tilde{\mathbf{v}}_\mathbf{c}$ and $\tilde{\mathbf{v}}_\mathbf{e}$. The 3D angular error between the $\tilde{\mathbf{v}}_\mathbf{c}$ and $\tilde{\mathbf{v}}_\mathbf{e}$ is:

$$\psi_E = \arccos(\hat{\mathbf{v}}_\mathbf{c} \cdot \hat{\mathbf{v}}_\mathbf{e}). \tag{5.4}$$

We can also compute the relative magnitude difference (ME) between the correct flow and the computed flow using:

$$\text{ME} = \frac{\|\vec{v}_c - \vec{v}_e\|}{\|\vec{v}_c\|} * 100\%, \tag{5.5}$$

and the direction difference (DE) as:

$$\text{DE} = \arccos\left(\hat{v}_c \cdot \hat{v}_e\right). \tag{5.6}$$

5.1.3.3 Error Analysis for 2D Optical Flow from Wedel et al.'s algorithm

We followed Fleet's method [3] to do quantitative analysis (average angular difference, relative magnitude difference, and average angular direction difference). We can write velocity as displacement per time unit as in $\vec{v} = (u, v)$ pixels/frame, or as a space-time direction vector $(u, v, 1)$ in units of (pixel, pixel, frame). Of course, velocity is obtained from the direction vector by dividing by the third component. Viewing and measuring velocity as orientation in space-time, it is natural to measure errors/differences as angular deviations from the correct space-time orientation. Therefore, we let velocities $\vec{v} = (u, v)^T$ be represented as 3D direction vectors[1], $\hat{v} \equiv \frac{1}{\sqrt{u^2+v^2+1}}(u, v, 1)^T$. The angular error between the correct velocity $\vec{v}_c$ and the measured velocity $\vec{v}_e$ is then:

$$\psi_E = \arccos(\hat{v}_c \cdot \hat{v}_e) \ . \tag{5.7}$$

Because of normalization, this error metric takes into account both direction and magnitude error. This error measure is convenient because it handles large and very small speeds without the amplification inherent in a relative measure of vector differences. It does have some bias however. For example, directional errors at small speeds do not give as large an angular error as similar directional errors at higher

[1]Note that the v is for 2D velocities written as $(u, v, 1)$ while $\mathbf{v}$ is for 3D velocities written as $(U, V, W, 1)$.

speeds [3].

5.1.4 Least Squares Range Flow Results

In this section, we present quantitative and qualitative results from the least squares algorithm.

5.1.4.1 Tabular Results for Least Squares

The Table 5.1 shows the 3D average angular error for U, V, W as well as the 3D magnitude error and the 3D direction error from execution of the least squares algorithm for different β values at the lowest pyramid level. Note that we used the word "error" but because we are not 100% confident in our ground truth they are really "differences". We used 4 pyramid levels and a downsampling of $\eta = 0.5$. When β is zero, the intensity information is not used. Non-zero β values weigh the depth and intensity contribution to the computed flow. When β is not a constant such as 10^{-3}, it is a local ratio. According to Spies et al. [10] β ratio should be computed in small local neighbourhoods as in Equation (3.34) in Chapter 3. This β ratio is different for every pixel location. The advantage of using a ratio for β values is that one doesn't have to guess the optimal β value using constant number. Usually (but not always) this provides among the best results.

STD in Table 5.1 is the abbreviation for standard deviation, while AAE is the abbreviation for average angular error, ME is the relative magnitude error and DE is the average angular direction error.

We observe from Table 5.1 that when the β values are ≤ 0.1 or the ratio is used as β, the AAE and ME are lower. However, when the β value is 1, the AAE, ME and DE are much higher. This is because the intensity derivatives are much larger than the depth derivatives so when β is 1, the least squares range flow operation is heavily

dominated by intensity derivatives, which really throw off the solution (especially for W). In order to obtain the optimal solution, we need a β value that scales the Z and I derivatives to be on the same order of magnitude (so that both constraints have roughly equal influence on the resulting 3D flow).

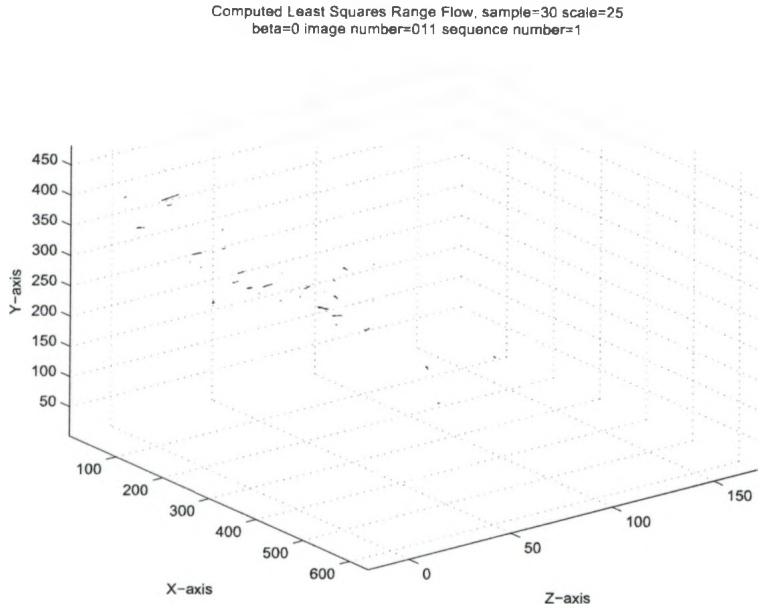
β	3D AAE ± STD	3D ME ± STD	3D DE ± STD
0	14.90°±9.73°	73.65 %±43.96%	6.84°±21.94°
10^{-3}	14.90°±9.73°	73.65 %±43.96%	6.84°±21.94°
10^{-2}	14.90°±9.73°	73.65 %±43.96%	6.83°±21.92°
10^{-1}	15.01°±9.72°	73.75 %±44.24%	7.21°±22.81°
1	28.34°±26.55°	248.40 %±423.91%	36.18°±56.24°
ratio	14.69°±9.72°	72.45 %±43.73%	7.97°±22.20°

Table 5.1: The 3D average angular difference, magnitude difference and the direction difference from the least squares range flow algorithm with different β values for the 11^{th} image of Sequence 1.

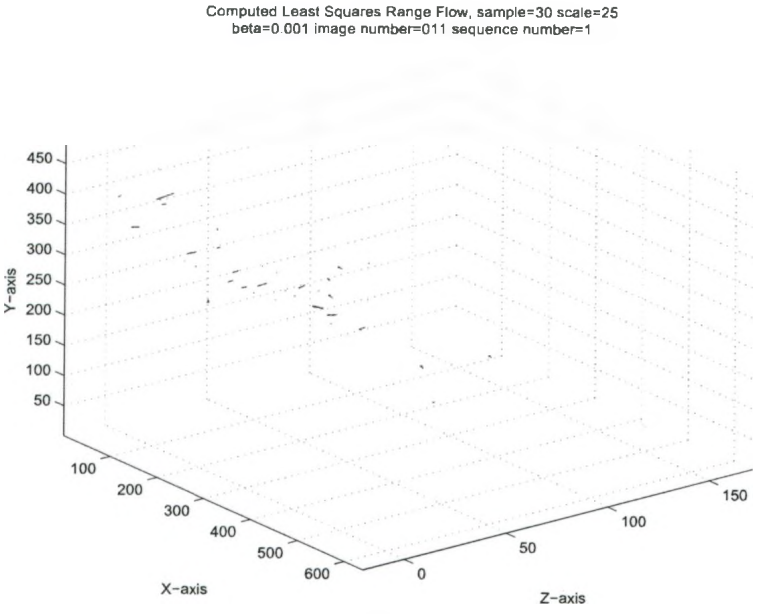
5.1.4.2 Graphical Results for Least Squares Algorithm

In the figures below, we show the graphical results computed from the least squares algorithm at the lowest level of the pyramid. We used a downsampling rate of $\eta = 0.5$ and 4 pyramid levels. We see that when $\beta = 1$, the qualitative answer looks much worse, which is consistent with the results in Table 5.1 because when $\beta >= 0.1$, the average angular difference is also higher when compared to the other β values. We think that bigger β values produce higher average angular difference because the ratio of the image gradients to the depth gradients becomes larger.

From the correct 3D flow field shown in Figure 5.3, we see that the best flow are where the most significant disparities are. The least squares algorithm could not detect them because the least squares integration matrices at these locations were singular (The L_2 condition number was greater than 10^7).

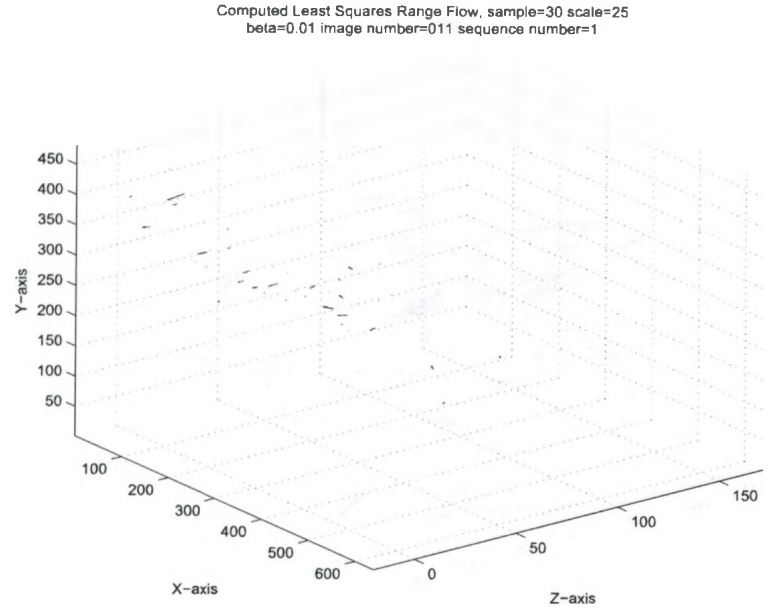


(a)

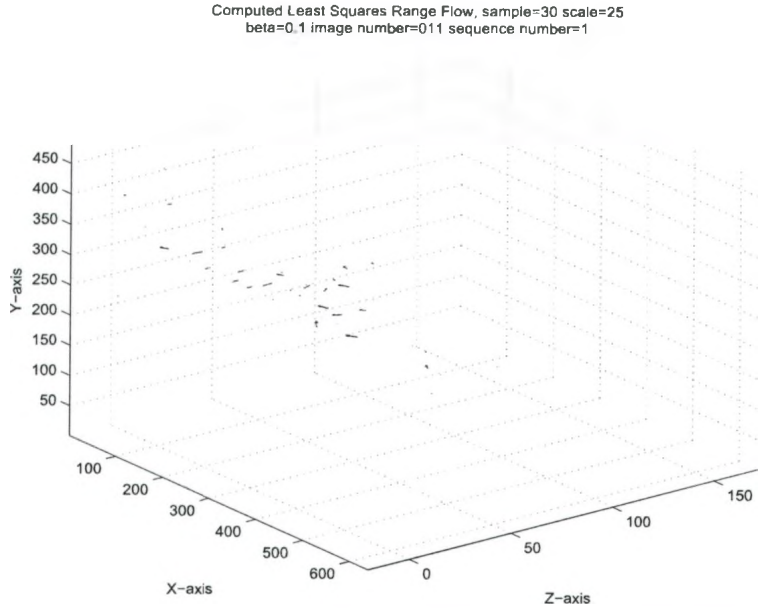


(b)

Figure 5.5: Least Squares Range Flow for the 11^{th} *image of Sequence* 1 *(a) for* $\beta = 0.0$ *where the AAE is 14.90°±9.73°, the ME is 73.65% ± 43.96% and the DE is 6.84°±21.94° and (b) for* $\beta = 0.001$ *, where the AAE is 14.90°± 9.73°, the ME is 73.65%± 43.96% and the DE is 6.84°± 21.94°.*

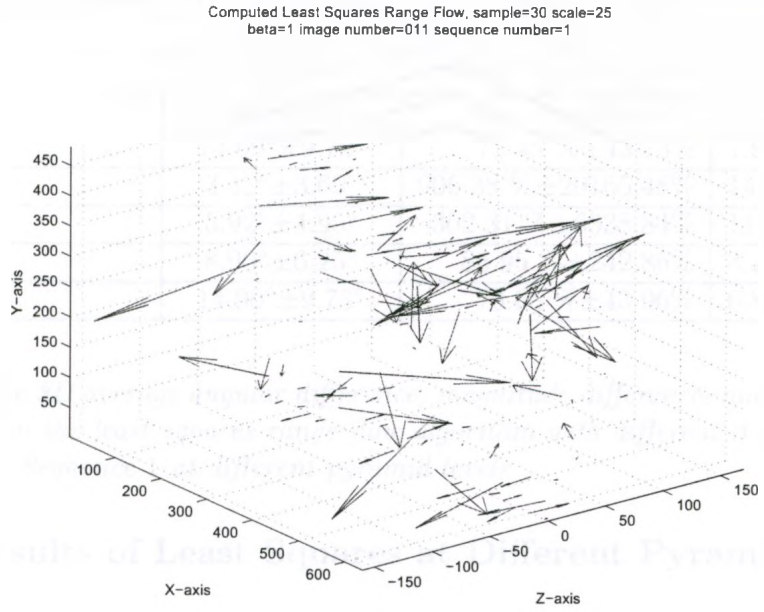


(c)

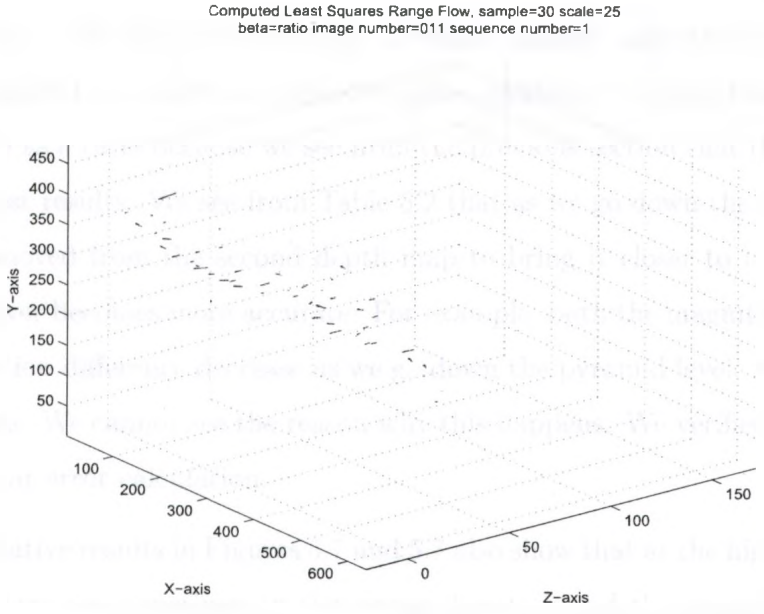


(d)

Figure 5.5: Least Squares Range Flow for the 11^{th} *image of Sequence* 1 *(c) for* $\beta = 0.01$ *where the AAE is 14.90°±9.73°, the ME is 73.65 %±43.96% and the DE is 6.83°±21.92° and (d) for* $\beta = 0.1$*, where the AAE is 15.01°±9.72°, the ME is 73.75 %±44.24% and the DE is 7.21°±22.81°.*



(e)



(f)

Figure 5.5: Least Squares Range Flow for the 11^{th} *image of Sequence* 1 *(e) for* $\beta = 1.0$ *where the AAE is 28.34°±26.55°, the ME is 248.40 %±423.91% and the DE is 36.18°±56.24° and (f) for* $\beta = 0.1$*, where the AAE is 14.69°±9.72°, the ME is 72.45 %±43.73% and the DE is 7.97°±22.20°.*

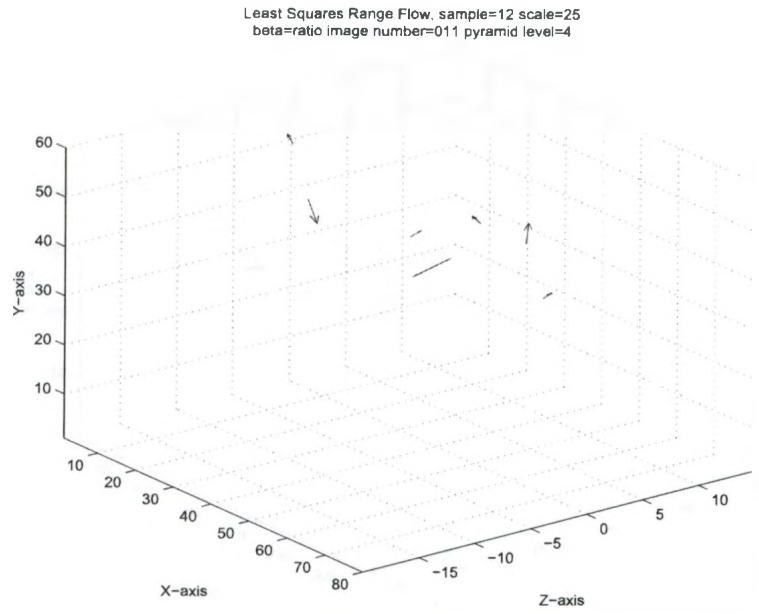
β	Pyramid Level	3D AAE ±STD	3D ME ±STD	3D DE ±STD
ratio	4	3.43°±3.72°	912.17 %±20351.76%	14.49°±35.94°
ratio	3	5.90°±4.96°	294.62 %±6345.18%	11.56°±27.45°
ratio	2	8.96°±6.29°	88.25 %±254.54%	9.10°±25.54°
ratio	1	14.69°±9.72°	72.45 %±43.73%	7.97°±22.20°
0	4	3.42°±3.69°	906.38 %±20165.48%	14.57°±36.04°
0	3	5.92°±4.95°	302.31 %±6338.84%	11.32°±26.89°
0	2	8.90°±6.25°	86.95 %±242.86%	8.20°±24.36°
0	1	14.90°±9.73°	73.65 %±43.96%	6.84°±21.94°

Table 5.2: The 3D average angular difference, magnitude difference, and the direction difference from the least squares range flow algorithm with different β values for the 11^{th} image of Sequence 1 at different pyramid levels.

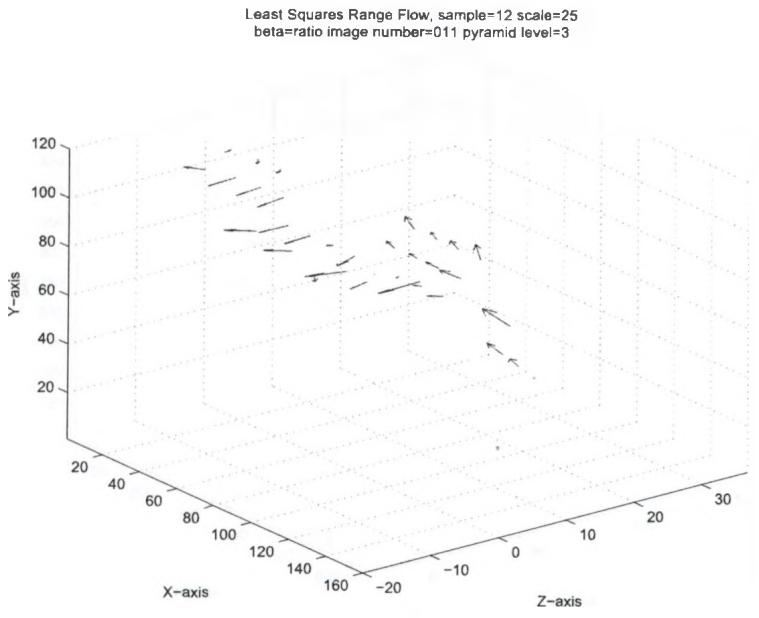
5.1.5 Results of Least Squares at Different Pyramid Levels

In this section, we present the results of the least squares algorithm at different pyramid levels. We show how putting the least squares algorithm in a pyramid structure improved the result as we go down the pyramid. We show these results for $\beta = 0.0$ and β as a ratio because we see from the previous section that these values of β gave the best results. We see from Table 5.2 that as we go down the pyramid level and W is removed from the second depth map to bring it closer to the first depth map, the answer becomes more accurate. For example, both the magnitude difference and the direction difference decrease as we go down the pyramid level. Strangely, the AAE increases. We cannot see the reason why this happens. We verified our MatLab average angular error calculation.

The qualitative results in Figures 5.7 and 5.7 also show that at the highest pyramid level, the vectors are sometimes in the wrong direction and the magnitude is much larger. Also, the standard deviations at the highest levels are high, which means there are a lot of variations. However, at the lowest level, most of the vectors are in the correct direction and the magnitude difference is smaller for both the β values of zero and when β is a ratio.

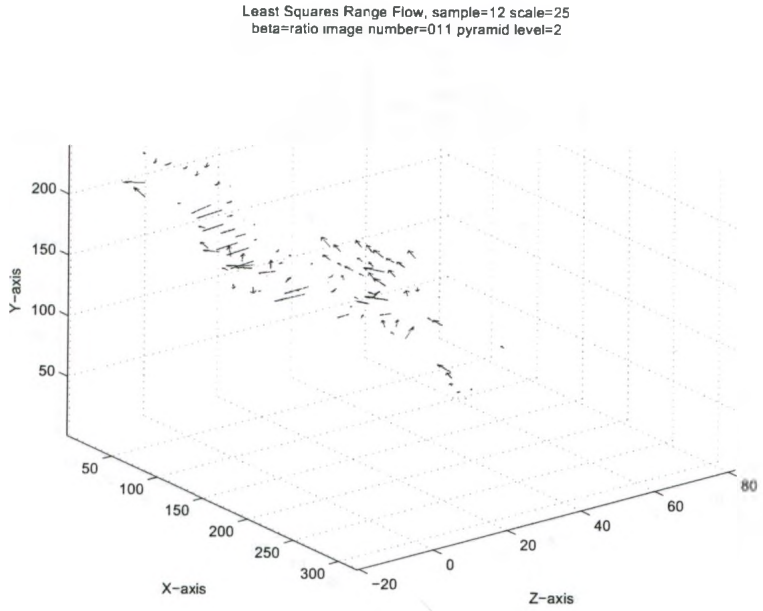


(a)

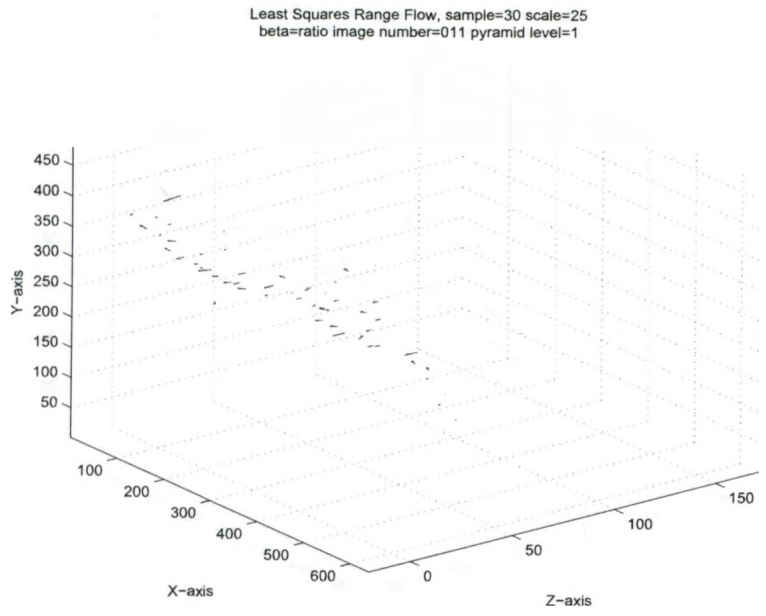


(b)

Figure 5.6: Least Squares Range Flow with β as a ratio for the 11^{th} image of Sequence 1 (a) at level 4 where the ME is 912.87 %±20351.76% and the DE is 14.49°±35.94° and (b) at level 3 where the ME is 294.62 %±6345.18% and the DE is 11.56°±27.45°.

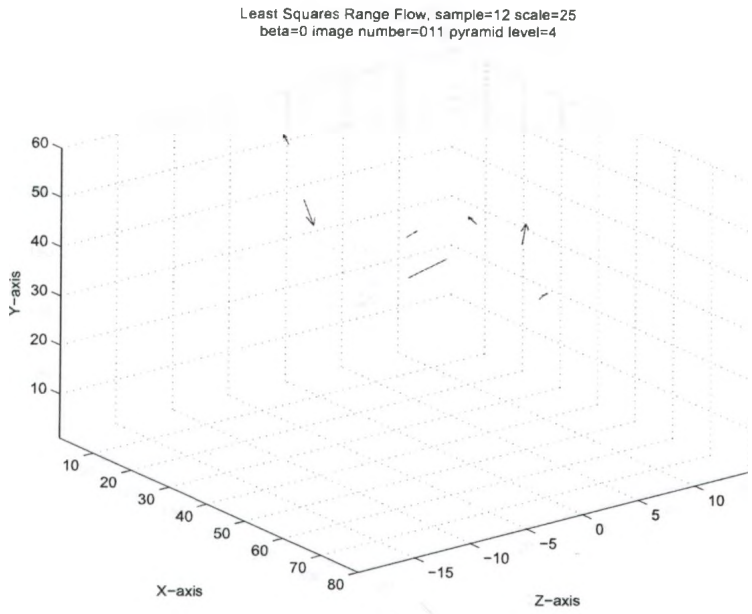


(c)

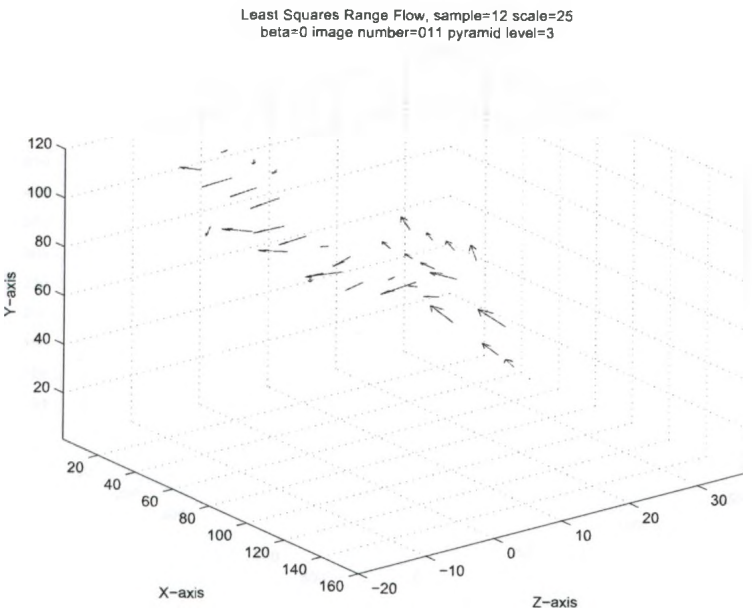


(d)

Figure 5.6: Least Squares Range Flow with β as a ratio for the 11^{th} image of Sequence 1 (c) at level 2 where the ME is 88.25 %±254.54% and the DE is 9.10°±25.54° and (d) at level 1 where the ME is 72.45 %±43.73% and the DE is 7.97°±22.20°.

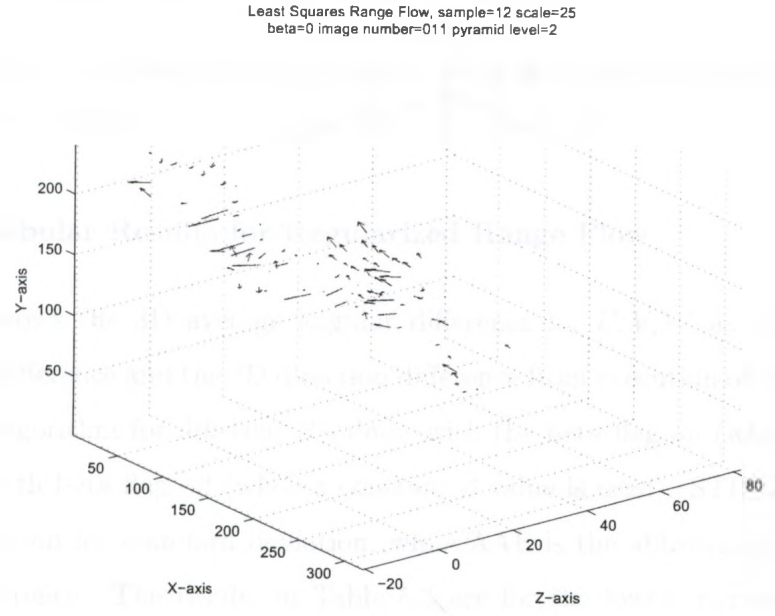


(a)

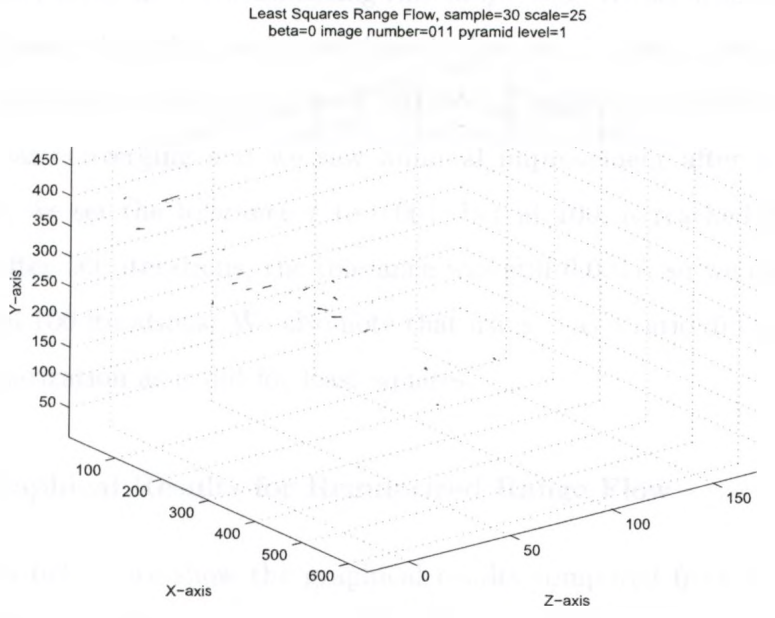


(b)

Figure 5.7: Least Squares Range Flow with $\beta = 0$ for the 11^{th} image of Sequence 1 (a) at level 4, where the ME is 906.38 %±20165.48% and the DE is 14.57°±36.04° and (b) at level 3, where the ME is 302.31 %±6338.84% and the DE is 11.32°±26.89°.



(c)



(d)

Figure 5.7: Least Squares Range Flow with $\beta = 0$ for the 11^{th} image of Sequence 1 (c) at level 2, where the ME is 86.95 %±242.86% and the DE is 8.20°±24.36° and (d) at level 1, where the ME is 73.65 %±43.96% and the DE is 6.84°±21.94°.

5.1.6 Regularized Range Flow Results

In this section, we present quantitative and qualitative results from the regularized range flow algorithm.

5.1.6.1 Tabular Results for Regularized Range Flow

Table 5.3 shows the 3D average angular difference for U, V, W as well as the 3D magnitude difference and the 3D direction difference from execution of the regularized range flow algorithm for different β values, with the beta flag on (when the ratio is used) and with beta flag off (when a constant β value is used). STD in Table 5.3 is the abbreviation for standard deviation, while AAE is the abbreviation for average angular difference. The results in Table 5.3 are for the lowest pyramid level. We used 4 pyramid levels and downsampling rate of $\eta = 0.5$. We see from Table 5.3 that the regularization algorithm performed based when the β value is set to zero and no intensity information is used. We used 100 for the maximum number of iterations because it was converging and we saw minimal improvement after 100 iterations. For example, we set the tolerance τ to 0.001, but at 100, it reached tolerance τ of 0.0021 but after 500 iterations, the tolerance was still 0.0017, so we decided just to use maximum 100 iterations. We also note that using β as a ratio did not perform as well for regularization as it did for least squares.

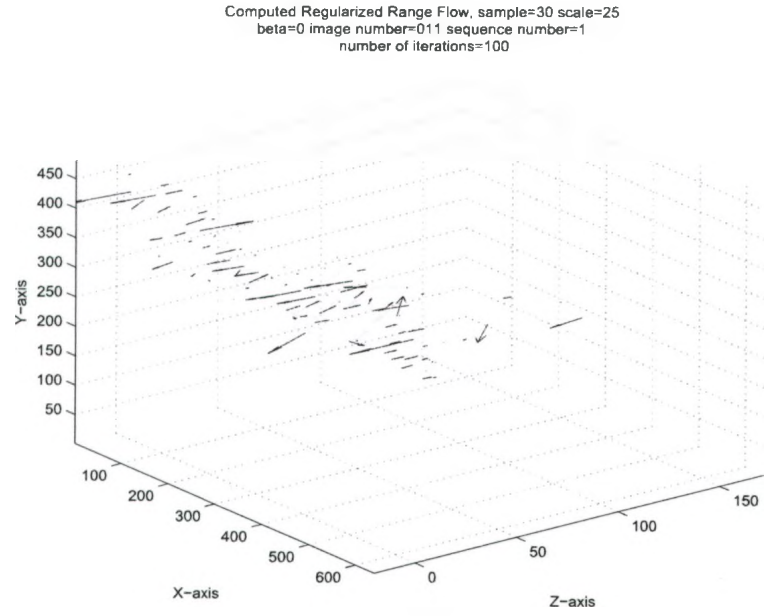
5.1.6.2 Graphical Results for Regularized Range Flow

In the figures below, we show the graphical results computed from the regularized algorithm. We see that when $\beta >= 0.01$, the qualitative results look really bad, which is consistent with the results of Table 5.3 because when $\beta >= 0.01$, the average angular difference is also high compared to the other β values. We think that bigger β values produce higher average angular difference because the ratio of the image gradient to the depth maps become higher.

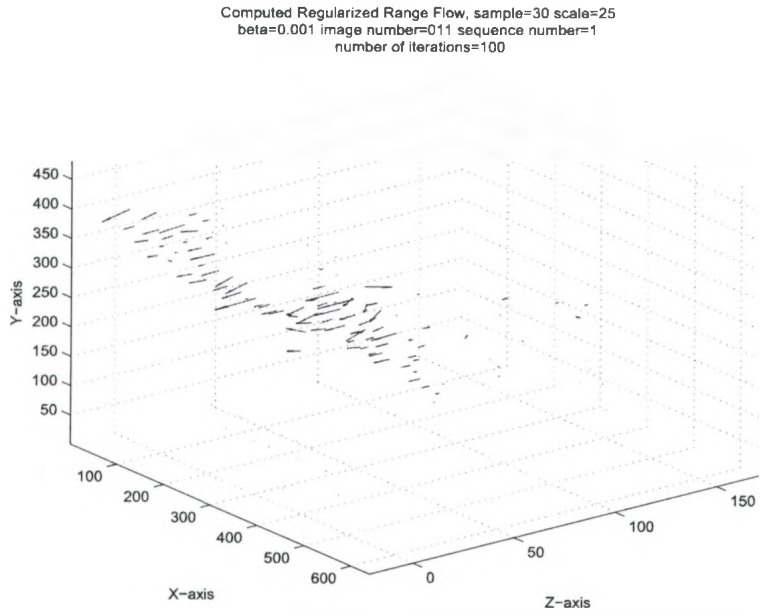
β	3D AAE ±STD	3D ME ±STD	3D DE ±STD
0.0	14.79°±10.40°	72.48 %±50.46%	16.58°±26.88°
10^{-3}	14.81°±10.40°	72.55 %±50.48%	16.43°±26.91°
10^{-2}	15.49°±10.42°	76.91 %±51.08%	19.97°±38.90°
10^{-1}	15.11°±9.32°	76.13 %±48.70%	34.27°±43.22°
1	26.92°±23.10°	179.42 %±216.29%	28.99°±40.29°
ratio	16.56°±11.78°	83.99 %±63.73%	35.55°±45.52°

Table 5.3: The 3D average angular difference, magnitude difference, and the direction difference from the regularization range flow algorithm with different β values for the 11^{th} image of Sequence 1.

Although in the correct 3D flow field shown in Figure 5.3 we see that the best flow are where the most significant disparities are, the regularization algorithm did not detect them because the inverse matrix at that location is singular (thresholding on the condition number 10^6 eliminated them). We expected the opposite.

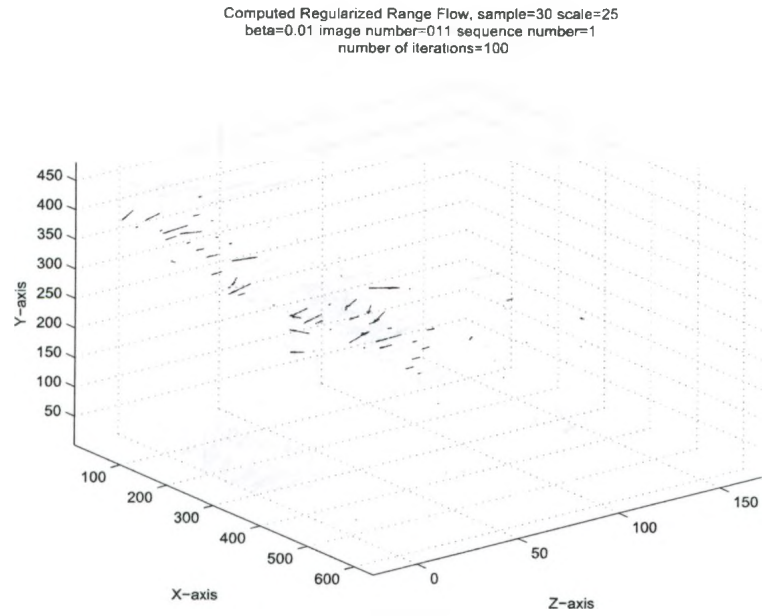


(a)

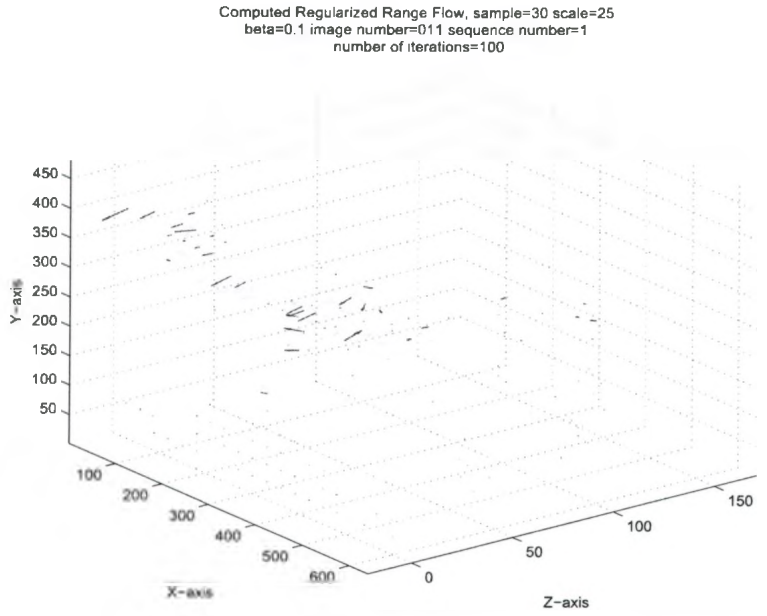


(b)

Figure 5.8: Regularized Range Flow for the 11^{th} *image of Sequence* 1 *(a) for* $\beta = 0.0$ *where the AAE is 14.79°±10.40°, the ME is 72.48 %±50.46%, and the DE is 16.58°±26.88° and (b) for* $\beta = 0.001$*, where the AAE is 14.81°±10.40° , the ME is 72.55 %±50.48% and the DE is 16.43°±26.91°.*

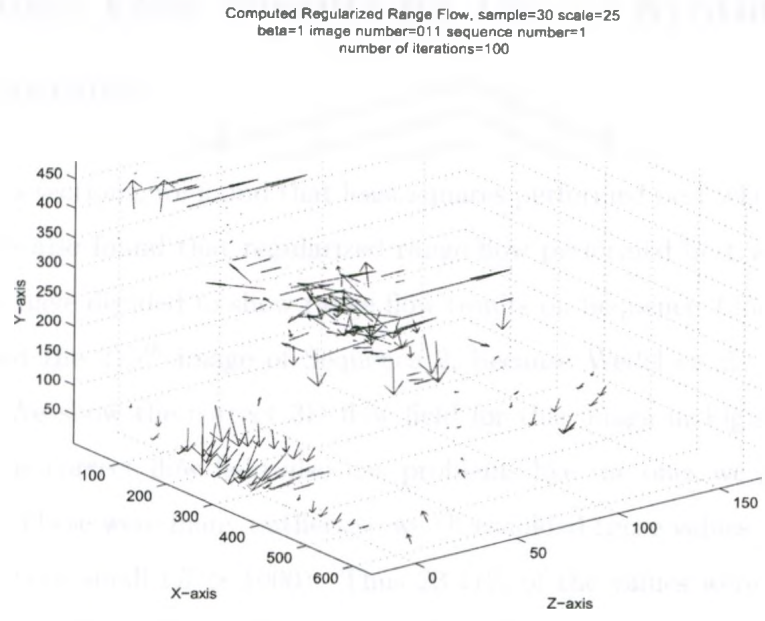


(c)

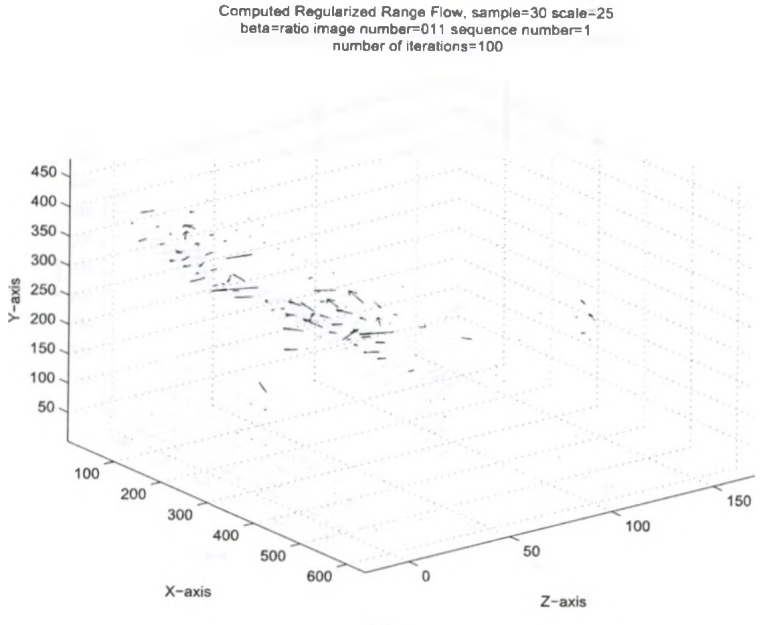


(d)

Figure 5.8: Regularized Range Flow for the 11^{th} *image of Sequence* 1 *(c) for* $\beta = 0.01$*, where the AAE is 15.49°±10.42°, the ME is 76.91 %±51.08% and the DE is 19.97°±38.90° and (d) for* $\beta = 0.1$*, where the AAE is 15.11°±9.32° , the ME is 76.13 %±48.70% and the DE is 34.27°±43.22°.*



(e)



(f)

Figure 5.8: Regularized Range Flow for the 11^{th} *image of Sequence* 1 *for (e)* $\beta = 0.1$*, where the AAE is 26.92°±23.10°, the ME is 179.42 %±216.29% and the DE is 28.99°±40.29° and (f)* β *as a ratio, where the AAE is 16.56°±11.78°, the ME is 83.99 %±63.73% and the DE is 35.55°±45.52°.*

5.2 Range Flow Results for the 2^{nd} Synthetic Car Sequence

In the previous sections, we found that least squares performed best when β was used as a ratio. We also found that regularized range flow performed best when $\beta = 0.0$. Therefore, we have decided to show range flow results on Sequence 2 for these values of β. We used the 215^{th} image of Sequence 2, because Wedel et al. used that in their paper. We show the correct 3D flow field for this image in Figure 5.9 below. We believe this correct flow field also has problems like the ones we mentioned in Section 5.1.1. There were many outliers so we thresholded those values whenever the disparity was very small ($Z > 1000$). Thus 23.41% of the values were thresholded. Notice that the 2 areas with moving cars have no correct flow.

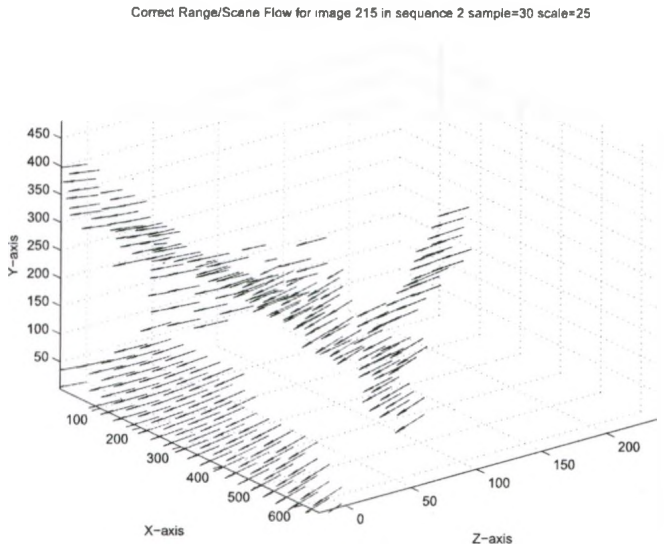


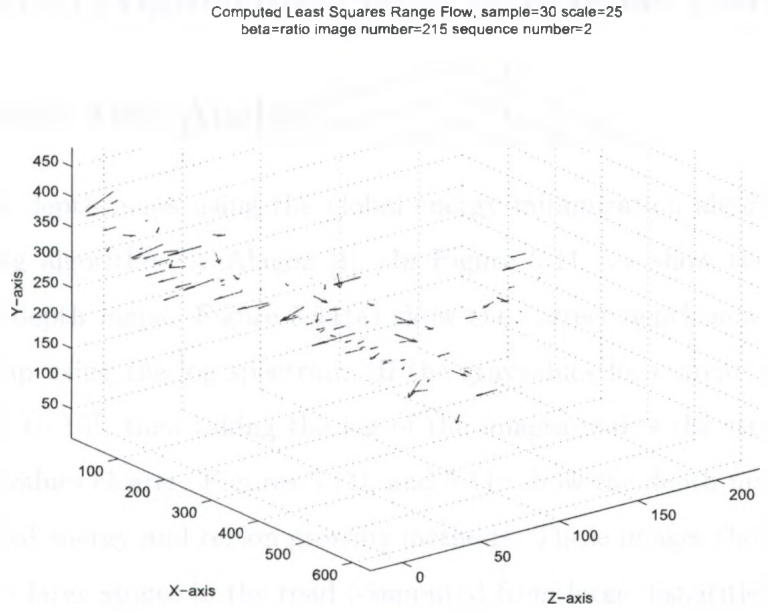
Figure 5.9: The correct 3D flow field for the 215^{th} image of Sequence 2.

We found that regularization converged at a much slower rate than for Sequence 1. For example, we specified a tolerance of 0.001 but found that after 100 iterations, the tolerance achieved was 0.007 and not 0.001. Therefore, we decided to let the

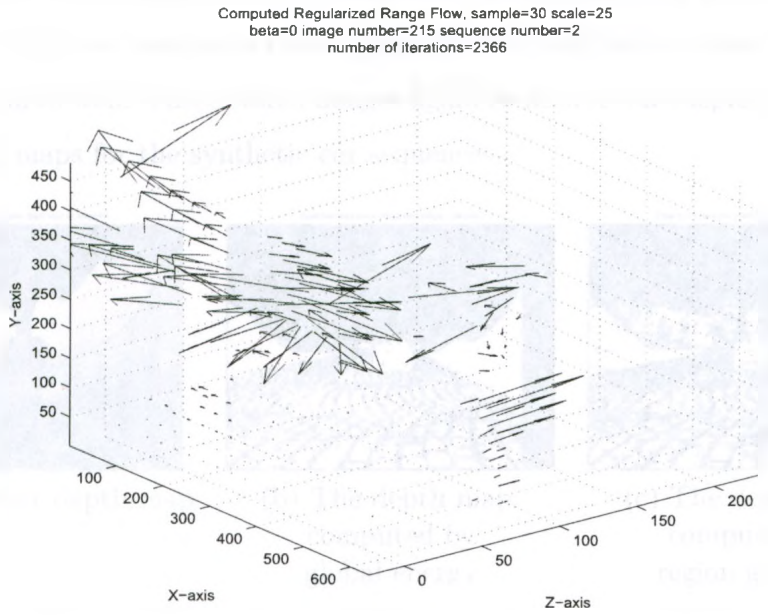
β	Algorithm	3D AAE ± STD	3D ME ± STD	3D DE ± STD
ratio	least squares	31.24°±20.94°	85.72 %±853.52%	12.30°±28.51°
0	regularization	41.43°±23.10°	154.19 %±4280.33%	26.93°±41.31°

Table 5.4: The 3D average angular difference, magnitude difference and the direction difference from the least squares range flow algorithm and regularized range flow for the 215^{th} image of Sequence 2.

program run for maximum number of iterations of 5000. We saw that tolerance was achieved at iteration 2336. This produced AAE of 41.43°±23.10°, the ME is 154.19 %±4280.33% and the DE is 26.93°±41.31° for $\beta = 0.0$. Table 5.4 shows that for least squares with β as ratio the AAE is 31.24°±20.94°, the ME is 85.72 %±853.52% and the DE is 12.30°±28.51°. We show the flow field in Figure 5.10.



(a)



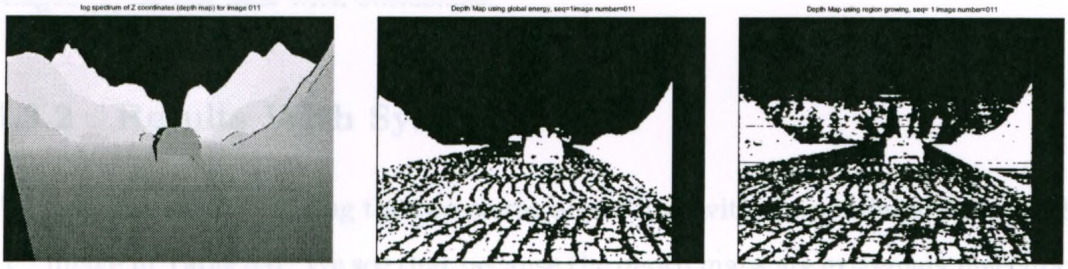
(b)

Figure 5.10: Range flow results for the 215^{th} *image of Sequence* 2 *for (a) Least Squares* β *as ratio, where the AAE is* $31.24^{\circ} \pm 20.94^{\circ}$, *the ME is* $85.72\% \pm 853.52\%$ *and the DE is* $12.30^{\circ} \pm 28.51^{\circ}$ *and (b) Regularized range flow using* β *as a ratio where the AAE of* $41.43^{\circ} \pm 23.10^{\circ}$, *the ME is* $154.19\% \pm 4280.33\%$ *and the DE is* $26.93^{\circ} \pm 41.31^{\circ}$.

5.3 Stereo Algorithms with Synthetic Car Data

5.3.1 Depth Map Analysis

We computed depth maps using the global energy minimization algorithm and the region growing algorithm by Alagoz [1]. In Figure 5.11, we show the correct and experimental depth maps. Figure 5.11(a) show the correct depth map. We viewed this depth map using the log spectrum. If the grayvalues have great variation in a image, say 10 to 10^7, then taking the log of the images makes the structure at the different grayvalues clearer. Figures 5.11b and 5.11c show the depth maps computed using the global energy and region growing methods. These images show the texture outlines of the large stones in the road (computed from large disparities) but do not show the tree and mountain depth outlines well (computed from small disparities). The correct depth map seems to have a plane as the road but captures the tree and mountain outlines well. These stereo images again re-inforce our suspicions about the correct depth maps for the synthetic car sequences.



(a) The correct depth map

(b) The depth map computed by global energy

(c) The depth map computed by region growing

Figure 5.11: (a) The correct versus (b) and (c) experimental depth maps generated by the global energy and region growing stereo algorithms by Alagoz [1] for the 11^{th} *image in Sequence 1.*

Type of Depth Map	Magnitude Difference ±STD
Depth Map using Global Energy	253.78 %±406.01%
Depth Map using Region Growing	211.16 %±375.28%

Table 5.5: The magnitude difference between Wedel et al's correct depth map and the depth maps computed using the stereo algorithms by Alagoz [1] for the 11^{th} image of Sequence 1

We show the magnitude difference the between the correct depth map and the depth maps computed by the stereo algorithms in Table 5.5. Table 5.5 shows that there is a high amount of error between Wedel et al.'s depth map and the computed depth maps. This is because the two stereo algorithms do not seems to capture the tree and mountain outlines well (these are far away objects with small disparities). Occlusions may also be causing some of the problems as well.

Table 5.5 shows the difference between Wedel et al.'s computed depth map and the depth maps computed by the stereo algorithm for the 11^{th} image of Sequence 1. We see that the magnitude difference is very high since the stereo algorithms by Alagoz [1] do not deal with occlusions.

5.3.2 Results With Synthetic Data

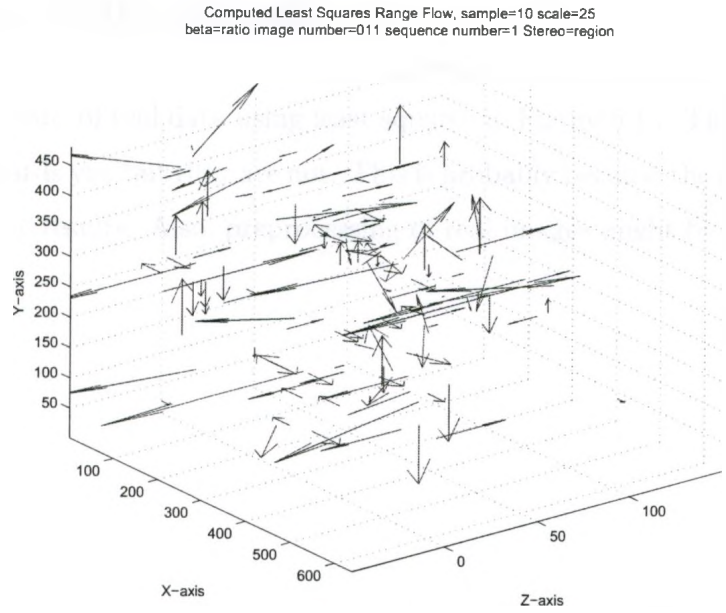
We show the results of using the two stereo algorithms with the synthetic data for the 11^{th} image in Table 5.6. We see that because the depth maps are extremely inaccurate as shown by the results in Table 5.5, the least squares results are inaccurate as shown in Table Table 5.6. Note that we have reduced the sampling from 30 (when using the correct depth map) to 10 when using computed depth maps from the Alagoz [1] stereo algorithms.

We also show the qualitative result of least squares with the β ratio on the 11^{th} image of Sequence 1 using the global energy and the region growing stereo algorithm in Figure 5.12. We see that the vectors are in the incorrect direction and the magnitudes

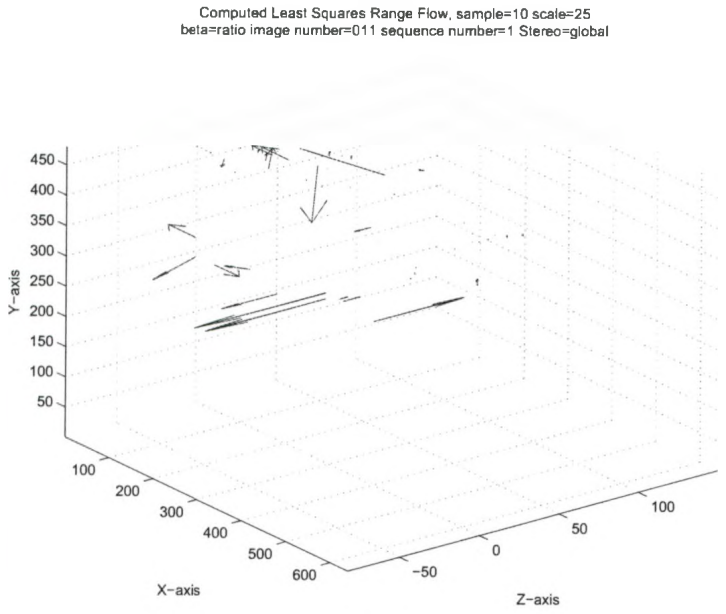
Type of Depth Map	β	3D Mag. Difference ±STD	3D Dir. Difference ±STD
Region Growing	ratio	125.50 %±152.83%	94.86°±43.04°
Global Energy	ratio	103.38 %±54.94%	89.63°±27.92°

Table 5.6: The 3D magnitude difference and direction difference between the correct range/scene flow and least squares range flow with depth maps computed from Alagoz [1] stereo algorithms for the 11^{th} *image of Sequence 1*

of the vectors are much larger. This may be caused by the outliers in the depth map. We need to put the least squares range flow in a robust framework to deal with outliers. The fact that there are such large differences in the depth values with respect to the correct depth map may also account for some of these problems.



(a)

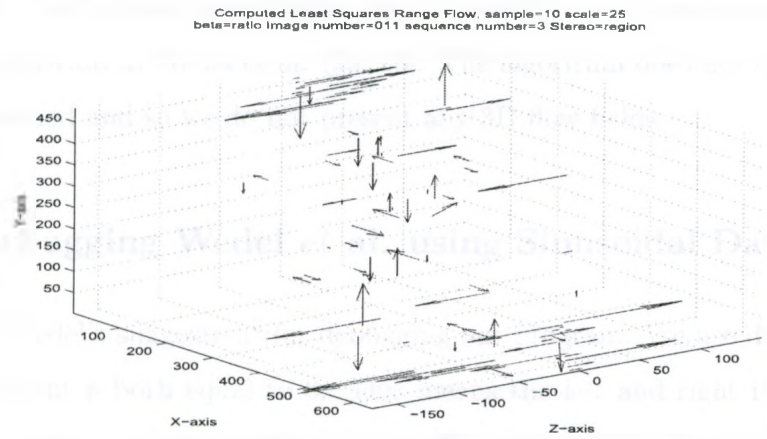


(b)

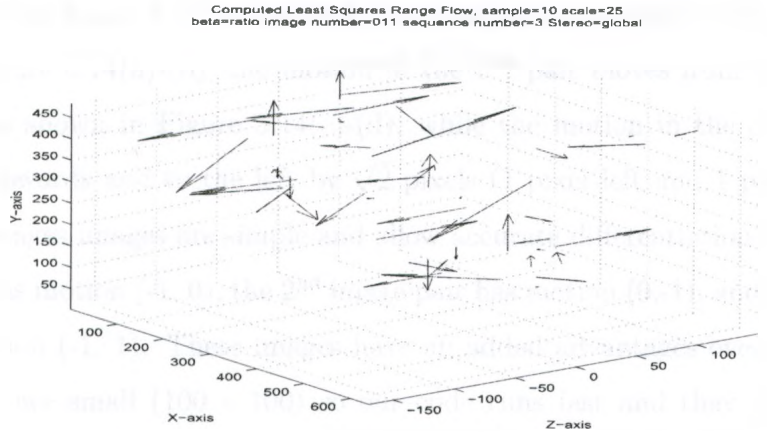
Figure 5.12: Least Squares range flow with β ratio for the 11^{th} image of Sequence 1 *(a) using computed depth maps from region growing algorithm where the magnitude difference is 125.50 %±152.83% and the direction difference is 94.86°±43.03° and (b) using computed depth maps from global energy algorithm where the magnitude difference is 103.38 %±54.94% and the direction difference is 89.63°±27.92°*

5.4 Real Data Result

We show the result of real data using least squares in Figure 5.13. The vectors should be coming towards you but they are not. This is probably because the stereo algorithm gives inaccurate results. Also, preprocessing of real images might be necessary.



(a)



(b)

Figure 5.13: Least Squares range flow with β ratio for the 11^{th} image of Sequence 3 (a) using computed depth maps from region growing algorithm and (b) using computed depth maps from global energy algorithm

5.5 Results from the Implementation of Wedel et al. Scene Flow Algorithm

In this section, we present some results we obtained from our implementation of the scene flow algorithm in Wedel et al. [29, 28]. The algorithm does not currently work as well as claimed and so we do not present any 3D flow fields.

5.5.1 Debugging Wedel et al. using Sinusoidal Data

We followed Wedel's suggestion[2] for debugging our program. We use disparity d and disparity gradient p both equal to 0. This means the left and right images are the same (so no stereo) and scene flow is simply 2D optical flow. We used 3 sets of 2D sinusoidal image pairs: the motion in 1^{st} pair moves from right to left by 1 pixel as shown in Figure 5.14(a)-(b); the motion in the 2^{nd} pair moves from bottom to top by 1 pixel as shown in Figure 5.14(c)-(d); while the motion in the 3^{rd} pair moves diagonally upwards and to the left by $\sqrt{2}$ pixels (1 pixel left and 1 pixel upwards). Sinusoidal images images are simple and allow accurate differentiation. Thus, the 1^{st} image pair has motion (-1, 0), the 2^{nd} image pair has motion (0,-1), and the 3^{rd} image pair has motion (-1, 1). These images have an added advantages over other images in that they are small (100×100) so our code runs fast and they allow accurate differentiation by most derivative kernels. The first two motion allows us to detect problems along the x and y dimensions.

[2]By email on June 17th,2011.

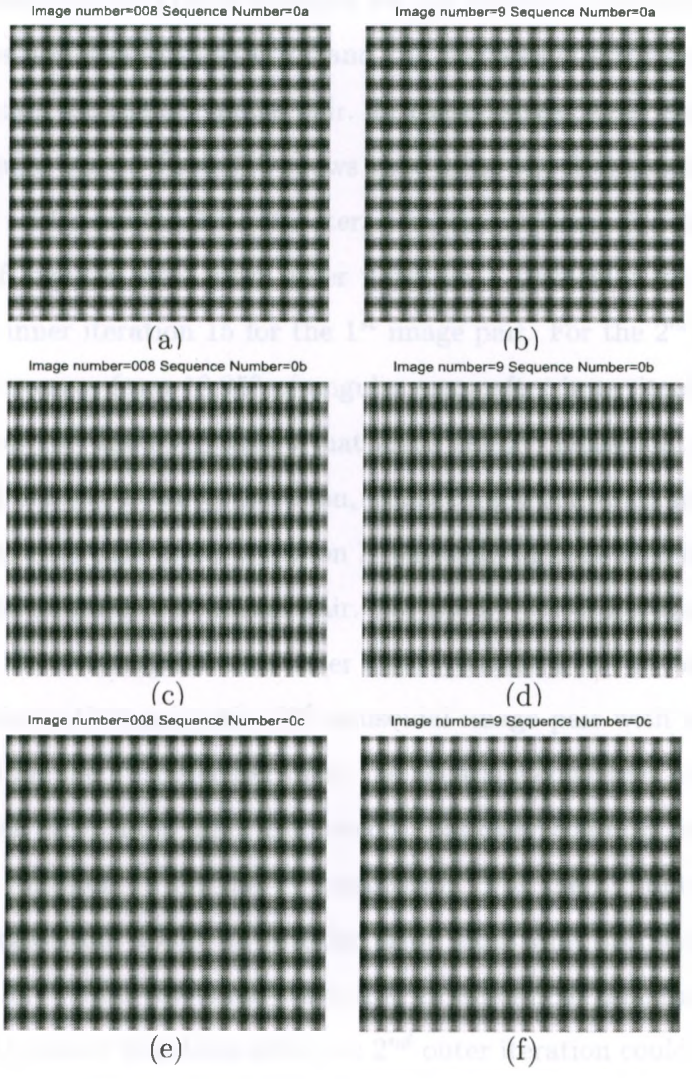


Figure 5.14: The 1^{st} sinusoidal pair in (a)-(b), the 2^{nd} sinusoidal pair in (c)-(d) and 3^{rd} sinusoidal image pair in (e)-(f).

We use 2 outer iterations and 15 inner iterations as suggested by Wedel et al. We view how the angular error and magnitude error change from inner and out iteration for the right to left motion in Table 5.7. We view how the angular error and magnitude error change from inner and outer iteration for the bottom to up motion in Table 5.8. Table 5.9 shows how the angular error and the magnitude error change from inner and outer iteration for the 3^{rd} image pair. In these tables, AAE is the abbreviation for average angular error. Table 5.7 shows that using the first sinusoidal image pair with motion of (-1,0), for the 1^{st} outer iteration, the accuracy converges nicely from about 44.31° of angular error after inner iteration 1 (basically the flow field is all 0's) to 4.58° at inner iteration 15 for the 1^{st} image pair. For the 2^{nd} outer iteration, the accuracy converges from 18.95° of angular error after inner iteration 1 to 11.36° at inner iteration 15. Table 5.8 shows that using the 2^{nd} sinusoidal image pair with motion of (0,-1), for the 1^{st} outer iteration, the accuracy converges nicely from about 43.96° of angular error after inner iteration 1 (basically the flow field is all 0's) to 2.70° at inner iteration 15 for the 1^{st} image pair. For the 2^{nd} outer iteration, the accuracy converges from 18.10° of angular error after inner iteration 1 to 7.09° at inner iteration 15. Table 5.9 shows that using the 3^{rd} sinusoidal image pair with motion of (-1,1), and we see that for the 1^{st} outer iteration, the accuracy converges nicely from about 53.93° of angular error after inner iteration 1 (basically the flow field is all 0's) to 11.49° at inner iteration 15 for the 1^{st} image pair. For the 2^{nd} outer iteration, the accuracy converges from 13.03° of angular error after inner iteration 1 to 9.39° at inner iteration 15. Wedel (by email) says that it is possible that the error after the convergence of the inner iterations after the 2^{nd} outer iteration could be greater than the the error after the convergence of the inner iterations after the 1^{st} outer iteration but it should not happen all the time (otherwise he suggests that something may be wrong with the warping). In our experience, the error is always greater after the 2^{nd} iteration than after the 1^{st} outer iteration. We believe our warping is correct. This is an open research problem for us.

outer iteration	inner iteration	2D AAE ±STD	2D ME ±STD	2D DE ±STD
1	1	44.31°±0.31°	98.79 %±0.55%	31.93°±21.32°
1	2	37.04°±3.14°	85.87 %±5.62%	32.11°±18.11°
1	3	23.71°±9.97°	59.24 %±15.69%	23.78°±16.02°
1	4	17.72°±9.04°	44.53 %±19.76%	18.18°±15.28°
1	5	13.73°±9.39°	34.91 %±20.69%	13.84°±14.67°
1	6	10.84°±9.61°	27.74 %±21.06%	10.71°±14.22°
1	7	8.73°±9.81°	22.25 %±21.47%	8.46°±13.90°
1	8	7.19°±9.98°	18.16 %±21.83%	6.83°±13.68°
1	9	6.11°±10.10°	15.23 %±22.11%	5.64°±13.50°
1	10	5.37°±10.18°	13.30 %±22.30%	4.76°±13.35°
1	11	4.90°±10.23°	12.18 %±22.40%	4.10°±13.23°
1	12	4.64°±10.25°	11.70 %±22.41%	3.59°±13.13°
1	13	4.53°±10.25°	11.70 %±22.34%	3.20°±13.05°
1	14	4.53°±10.22°	11.96 %±22.22%	2.90°±12.97°
1	15	4.58°±10.19°	12.34 %±22.08%	2.66°±12.91°
2	1	18.95°±7.10°	91.56 %±18.01%	3.90°±10.41°
2	2	16.21°±7.65°	66.96 %±16.91%	7.35°±10.51°
2	3	15.03°±8.09°	57.93 %±18.73%	8.36°±11.33°
2	4	14.44°±8.30°	54.48 %±19.26%	8.30°±11.77°
2	5	13.94°±8.48°	52.06 %±19.54%	7.96°±12.02°
2	6	13.54°±8.61°	50.15 %±19.66%	7.69°±12.20°
2	7	13.20°±8.70°	48.54 %±19.72%	7.46°±12.33°
2	8	12.91°±8.78°	47.14 %±19.77%	7.25°±12.43°
2	9	12.64°±8.84°	45.88 %±19.80%	7.06°±12.50°
2	10	12.39°±8.90°	44.74 %±19.84%	6.89°±12.57°
2	11	12.16°±8.96°	43.68 %±19.88%	6.73°±12.63°
2	12	11.95°±9.01°	42.70 %±19.92%	6.57°±12.67°
2	13	11.74°±9.05°	41.78 %±19.97%	6.42°±12.72°
2	14	11.55°±9.10°	40.90 %±20.03%	6.28°±12.76°
2	15	11.36°±9.14°	40.07 %±20.09%	6.14°±12.80°

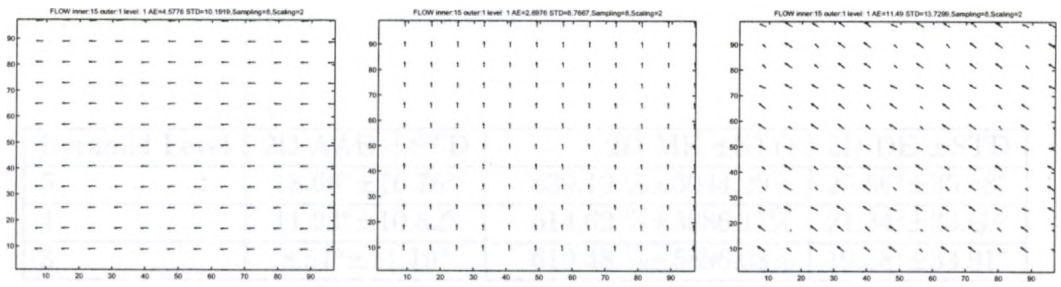
Table 5.7: The 2D average angular error, magnitude error, and the direction error from Wedel et al.'s algorithm for the 1st sinusoidal image pair with right to left motion.

outer iteration	inner iteration	2D AAE ±STD	2D ME ±STD	2D DE ±STD
1	1	43.96°±0.40°	98.20 %±0.69%	11.88°±8.36°
1	2	34.71°±3.53°	81.76 %±6.34%	12.33°±7.49°
1	3	19.19°±7.24°	49.90 %±14.88%	12.77°±7.14°
1	4	11.98°±9.04°	31.25 %±17.67%	10.47°±6.19°
1	5	8.32°±8.35°	21.56 %±18.58%	7.14°±5.35°
1	6	6.05°±8.60°	15.54 %±19.22%	4.72°±4.65°
1	7	4.65°±8.75°	11.79 %±19.62%	3.15°±4.08°
1	8	3.81°±8.82°	9.54 %±19.79%	2.18°±3.61°
1	9	3.31°±8.84°	8.26 %±19.85%	1.59°±3.24°
1	10	3.04°±8.84°	7.60 %±19.82%	1.21°±2.93°
1	11	2.90°±10.23°	7.29 %±19.75%	0.98°±2.67°
1	12	2.82°±8.80°	7.12 %±19.68%	0.83°±2.46°
1	13	2.77°±8.79°	7.00 %±19.63%	0.74°±2.29°
1	14	2.73°±8.77°	6.91 %±19.59%	0.67°±2.15°
1	15	2.70°±8.77°	6.83 %±19.57%	0.62°±2.04°
2	1	18.10°±5.94°	89.13 %±16.10%	0.98°±1.80°
2	2	14.75°±6.64°	62.53 %±17.64%	4.05°±3.99°
2	3	12.55°±7.26°	41.51 %±16.36%	9.41°±5.54°
2	4	13.10°±7.64°	38.78 %±17.60%	12.52°±6.60°
2	5	12.68°±7.87°	36.74 %±18.38%	12.42°±6.70°
2	6	11.91°±8.04°	34.27%±18.90%	11.58°±6.61°
2	7	11.16°±8.19°	31.98 %±19.29%	10.70°±6.51°
2	8	10.48°±8.30°	29.91 %±19.58%	9.87°±6.38°
2	9	9.84°±8.39°	28.02 %±19.81%	9.10°±6.25°
2	10	9.26°±8.47°	26.31 %±19.99%	8.39°±6.11°
2	11	8.73°±8.54°	24.76 %±20.15%	7.73°±5.97°
2	12	8.25°±8.59°	23.36 %±20.72%	7.12°±5.83°
2	13	7.82°±8.64°	22.11 %±20.36%	6.57°±5.70°
2	14	7.43°±8.67°	21.00 %±20.42%	6.09°±5.57°
2	15	7.09°±8.70°	20.03 %±20.46%	5.64°±5.43°

Table 5.8: The 2D average angular error, magnitude error, and the direction error from Wedel et al.'s algorithm for the 2nd sinusoidal image pair with bottom to top motion.

outer iteration	inner iteration	2D AAE ±STD	2D ME ±STD	2D DE ±STD
1	1	53.93°±0.47°	99.00 %±0.58%	33.73°±24.13°
1	2	46.46°±5.28°	89.57 %±6.73%	32.06°±23.41°
1	3	31.07°±15.98°	64.68 %±25.60%	28.69°±23.36°
1	4	24.74°±18.29°	50.40 %±30.59%	23.58°±20.61°
1	5	21.87°±17.75°	44.97 %±33.31%	19.18°±17.54°
1	6	19.80°±17.16°	41.34%±32.51%	15.94°±15.19°
1	7	18.17°±16.62°	38.49 %±31.78%	13.56°±13.42°
1	8	16.85°±16.13°	36.14 %±31.09%	11.78°±12.09°
1	9	15.75°±15.67°	34.16 %±30.41%	10.42°±11.06°
1	10	14.81°±15.26°	32.43 %±29.76%	9.34°±10.28°
1	11	13.98°±14.88°	30.90 %±29.14%	8.48°±9.68°
1	12	13.25°±14.54°	29.52 %±28.55%	7.78°±9.20°
1	13	12.61°±14.23°	28.27 %±28.00%	7.21°±8.82°
1	14	12.02°±13.96°	27.11 %±27.49%	6.72°±8.52°
1	15	11.49°±13.73°	26.03 %±27.02%	6.32°±8.28°
2	1	13.03°±10.80°	46.39 %±22.01%	6.60°±7.55°
2	2	11.24°±11.54°	34.47 %±24.70%	6.80°±7.86°
2	3	10.63°±11.65°	31.01 %±24.53%	6.65°±7.83°
2	4	10.32°±11.66°	29.26 %±23.86%	6.50°±7.82°
2	5	10.10°±11.67°	28.06 %±23.38%	6.41°±7.84°
2	6	9.93°±11.69°	27.14%±23.08%	6.32°±7.87°
2	7	9.80°±11.72°	26.42 %±22.90%	6.26°±7.91°
2	8	9.70°±11.74°	25.85 %±22.79%	6.20°±7.95°
2	9	9.61°±11.77°	25.39 %±22.72%	6.16°±7.98°
2	10	9.55°±11.79°	25.03 %±22.68%	6.12°±8.00°
2	11	9.49°±11.81°	24.74 %±22.66%	6.09°±8.03°
2	12	9.46°±11.83°	24.51 %±22.64%	6.07°±8.05°
2	13	9.43°±11.84°	24.33 %±22.63%	6.05°±8.07°
2	14	9.41°±11.86°	24.19 %±22.63%	6.03°±8.09°
2	15	9.39°±11.87°	24.08 %±22.62%	6.01°±8.11°

Table 5.9: The 2D average angular error, magnitude error, and the direction error from Wedel et al.'s algorithm for the 3rd sinusoidal image pair with diagonal motion.



(a) (b) (c)

Figure 5.15: (a) Computed flow for 1^{st} *image pair with motion (-1,0) and angular error* 4.58°, *(b) computed flow for* 2^{nd} *image pair with motion (0,-1) and angular error* 2.70°, *and the (c) computed flow for* 3^{rd} *image pair of motion (-1, 1) with angular error* 11.49°. *The flows are for inner iteration 15 and outer iteration 1.*

Figures 5.15(a), (b) and (c) show the 2D optical flow computed by Wedel et al. for the 1^{st}, 2^{nd}, and 3^{rd} sinusoidal image pairs respectively.

One may wonder if warping is done correctly. We assume the warping algorithm we used that was coded by Thomas Pock [7] is correct, because that was the same code used in Ali Aljohani's project, which involves a MatLab implementation of 2D Brox et al. optical flow [6]. We report the results of that implementation here. He used $\eta = 0.7$, 5 pyramid levels, 25 outer iteration and 30 inner iteration for the 11^{th} image of Sequence 1. Table 5.5.1 shows that using Pock's method for warping improved average angular error as we go down the pyramid. We also note that the magnitude error is very high. This was a problem also observed in our range flow algorithm. We show the qualitative result of 2D Brox in Figure 5.16.

Pyramid Level	2D AAE ±STD	2D ME ±STD	2D DE ±STD
5	18.66°±16.76°	639.19 %±3044.20%	25.56°±35.48°
4	11.23°±10.82°	614.62 %±3886.17%	21.34°±33.60°
3	8.51°±11.19°	619.48 %±5896.08%	19.58°±33.91°
2	7.50°±12.52°	912.67 %±16474.99%	17.98°±33.20°
1	7.07°±14.04°	1111.26 %±23801.55%	16.01°±31.45°

Table 5.10: The 2D angular error, magnitude error, and the direction error from the computation of 2D Brox at different pyramid levels for the 11^{th} image of Sequence 1.

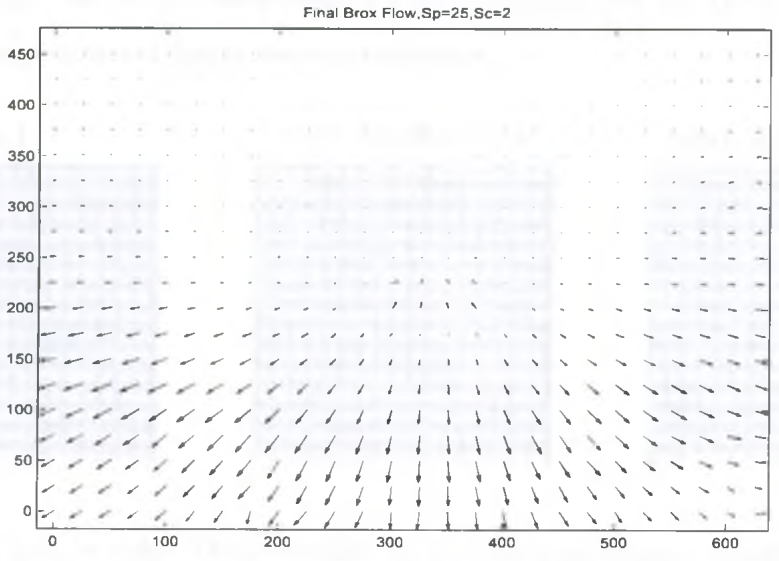


Figure 5.16: (a) The computed flow field (u, v) for the 11^{th} image of Sequence 1 using 2D Brox.

After outer iteration 1, the left and right images at time $t+1$ are warped back into time t: the computed motion is removed from the images. Figures 5.17, 5.18 and 5.19 show the original 1^{st} image, the original 2^{nd} image and the warped 2^{nd} image for the 3 sinusoidal sequences.

We can see from Figures 5.17, 5.18 and 5.19 that the average image difference after warping is 4 times less than the average image difference between the 1^{st} and 2^{nd} images. Now the inner iterations for the 2^{nd} outer iteration should compute the optical flow for the original images and the warped images (which should be small). At the end of the inner iteration this is added to the optical flow computed for outer iteration 1. For some unknown reason, our results after the second outer iteration are less accurate than after the first outer iteration. We also note that we start with a less accurate flow at the 1^{st} inner iteration of the 2^{nd} outer iteration than for the last inner iteration of the 1^{st} outer iteration. Since the warping seems to work we do not expect this. We get the same result for the 2^{nd} image pair but the 3^{rd} pair shows that warping produces a larger average difference.

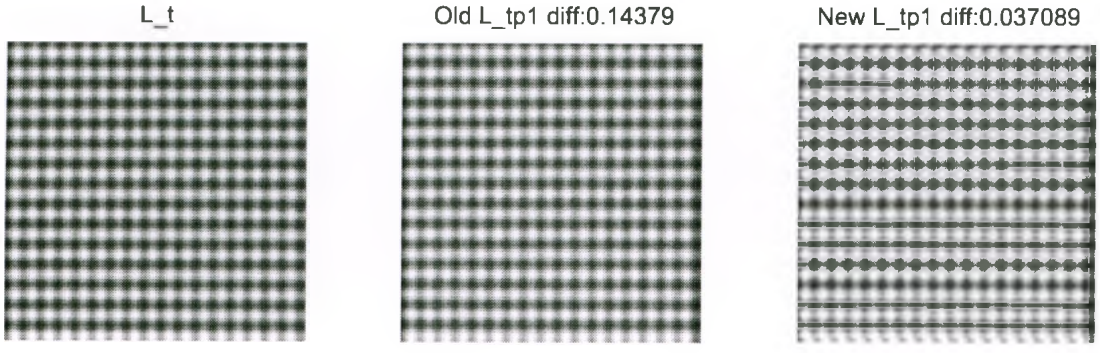


Figure 5.17: Left to right: The 1^{st} image, the 2^{nd} unwarped image (repeated here from Figure 5.14 for comparison purposes) and the 2^{nd} warped images. Printed above the images are the average difference with the 2^{nd} unwarped and warped images with the 1^{st} image. This shows how well warping works for the right to left motion.

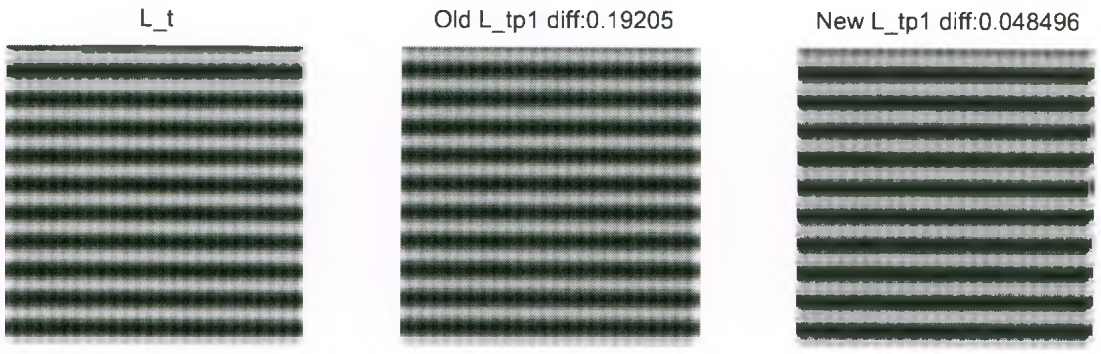


Figure 5.18: Left to right: The 1^{st} image, the 2^{nd} unwarped image (repeated here from Figure 5.14 for comparison purposes) and the 2^{nd} warped images. Printed above the images are the average difference with the 2^{nd} unwarped and warped images with the 1^{st} image. This shows how well warping works for the bottom to top motion.

5.5.2 Results of Wedel et al. on the Synthetic Car Data

We show the correct disparity, optical flow field in Figure 5.20(a)-(b) and the correct disparity gradient field in Figure 5.21 below for image 11 of Sequence 1, which was also used by Wedel et al. [28].

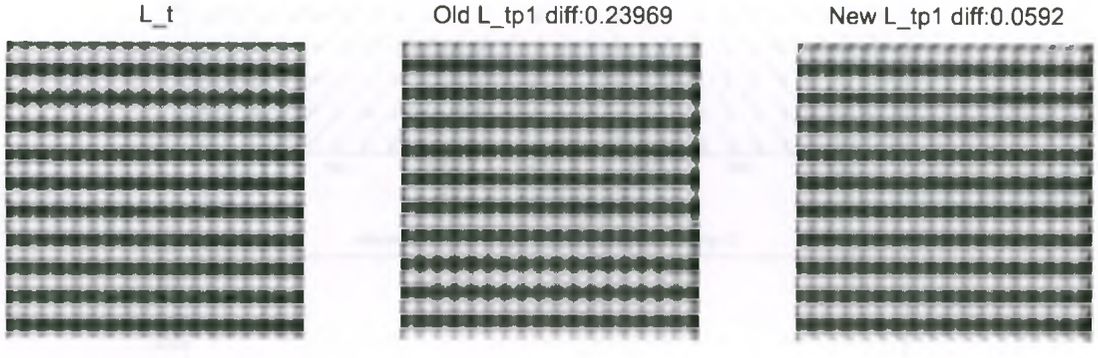


Figure 5.19: Left to right: The 1^{st} image, the 2^{nd} unwarped image (repeated here from Figure 5.14 for comparison purposes) and the 2^{nd} warped images. Printed above the images are the average difference with the 2^{nd} unwarped and warped images with the 1^{st} image. This shows how well warping works for the diagonal motion.

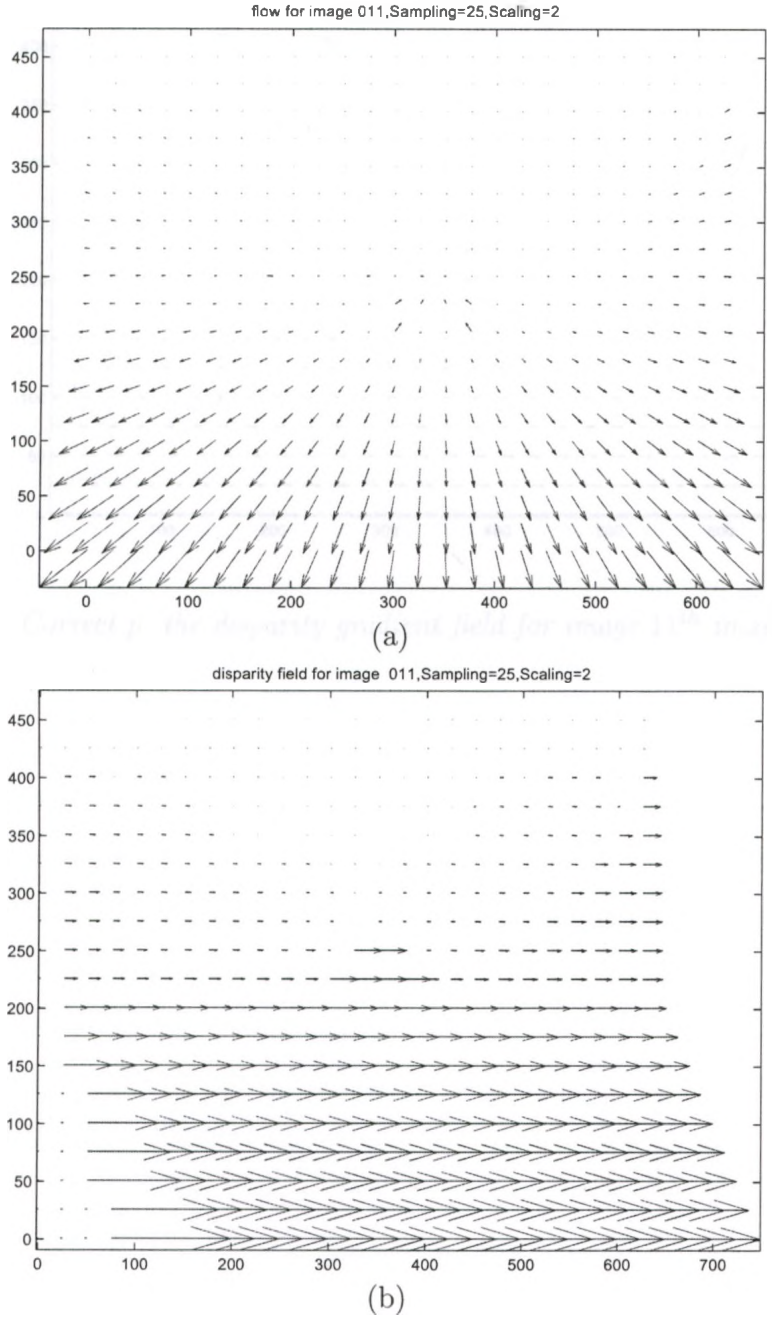


Figure 5.20: (a) Correct (u, v) and (b) Correct d for the 11^{th} image of Sequence 1

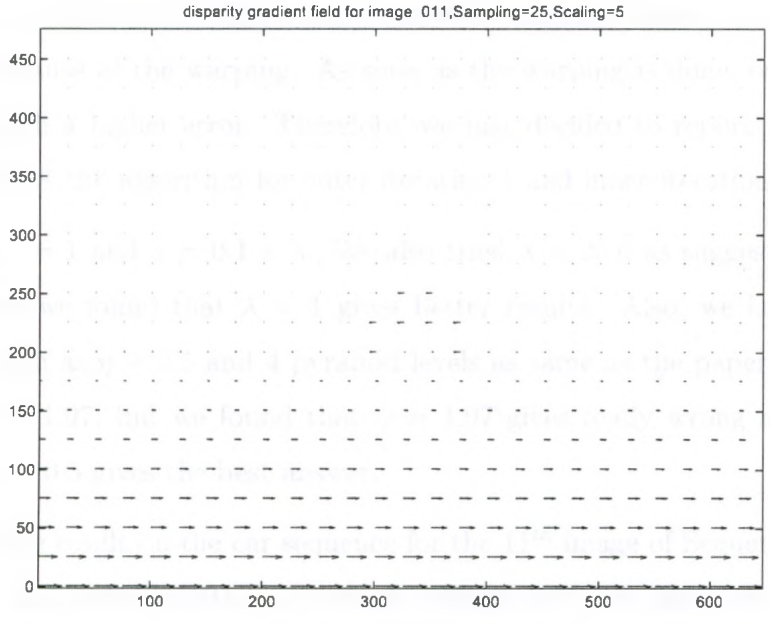


Figure 5.21: Correct p, the disparity gradient field for image 11^{th} *image of Sequence 1*

We saw in Section 5.5.1 that the solution gets worse after inner iteration 15, outer iteration 1 because of the warping. As soon as the warping is done, outer iteration 2 starts off with a higher error. Therefore, we just decided to report the result for Wedel et al's [28, 29] algorithm for outer iteration 1 and inner iteration 15.

We used $\lambda = 1$ and $\gamma = 0.1 \times \lambda$. We also tried $\lambda = 20.0$ as suggested by Wedel via email, but we found that $\lambda = 1$ gives better results. Also, we kept the other parameters such as $\eta = 0.5$ and 4 pyramid levels as same as the paper. And Wedel suggested $\omega = 1.97$, but we found that $\omega = 1.97$ gives really wrong result and we found that $\omega = 0.5$ gives the best answer.

We show the result on the car sequence for the 11^{th} image of Sequence 1 at inner iteration 15, and outer iteration 1 with the Wedel test (left and right image same and no disparity) and without the Wedel test (when left and right image are not the same and there is disparity and p, the gradient of disparity, is also computed). With the Wedel test, we got average angular error of $14.69° \pm 17.88°$, direction error of $38.69° \pm 41.03°$ and magnitude error of $417.50\% \pm 7156.032$ as shown in Figure 5.22. Without the Wedel test, we got average angular error of $18.13° \pm 20.01°$, direction error of $46.77° \pm 43.30°$ and magnitude error of $681.74\% \pm 9844.14$ as shown in Figure 5.23. We see that the error measure for Figure 5.23 is higher than Figure 5.22 after we turned off the Wedel test (when disparity was included). We believe this is happening because error for p is very high. For example, we can see from Figure 5.24 that the computed p has much higher magnitude and is much longer than the correct disparity gradient field shown in Figure 5.21. It also has many outliers.

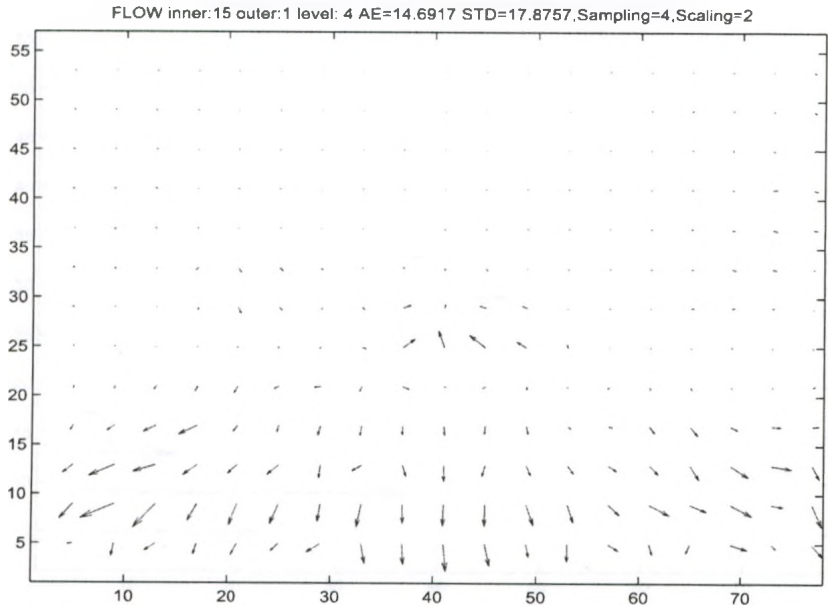


Figure 5.22: The computed flow field 11^{th} image in Sequence 1 with average angular error of $14.69° \pm 17.88°$ using Wedel test ($d = p = 0$).

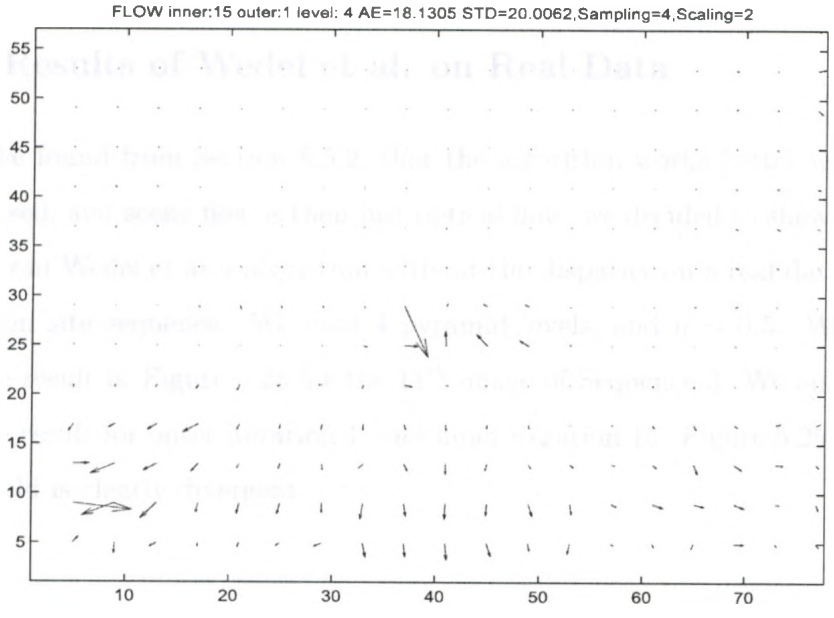


Figure 5.23: The computed flow field for the 11^{th} image of Sequence 1 with average angular error of $18.13° \pm 20.01°$ without using Wedel test, d and p not zero.

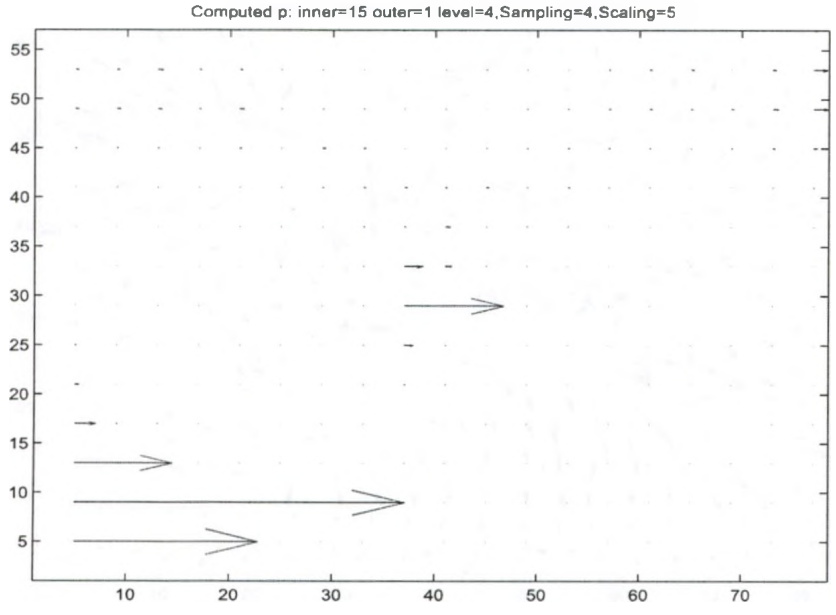


Figure 5.24: The computed disparity gradient field p for the 11^{th} image of Sequence 1.

5.5.3 Results of Wedel et al. on Real Data

Because we found from Section 5.5.2, that the algorithm works better when no disparity is used, and scene flow is then just optical flow, we decided to show the optical flow u, v from Wedel et al.'s algorithm without the disparity on a real data called the construction site sequence. We used 4 pyramid levels, and $\eta = 0.5$. We show the qualitative result in Figure 5.25 for the 11^{th} image of Sequence 3. We only show the qualitative result for outer iteration 1, and inner iteration 15. Figure 5.25 shows that the flow field is clearly divergent.

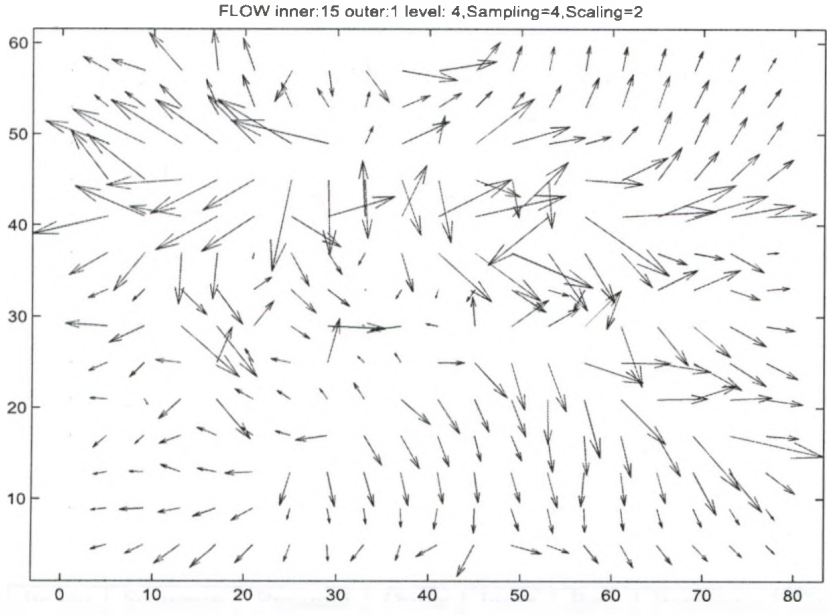


Figure 5.25: The computed flow field (u, v) for the 11^{th} image of Sequence 3.

5.6 Running Time

Table 5.11 presented the time required for our algorithm running on the 11_{th} image of Sequence 1. We ran our program on newfie.csd.uwo.ca, which is a Linux (Fedora Core 9.3) machine at the University of Western Ontario, Computer Science department. It is a 64-bit machine with 2.8GHz clock speed and 8GB of main memory. The time is given in seconds. For the 11^{th} image of Sequence 1, the least squares algorithm was the most time efficient with 30.89 seconds. The slowest was scene flow.

Algorithm	Image	Sequence	Pyramid Level	Outer Iter.	Inner Iter.	Beta	Iterations Used	Time
Scene Flow	11^{th}	1	4	2	15	N/A	N/A	1212.15 sec.
Least Squares	11^{th}	1	4	N/A	N/A	ratio	100	30.89 sec.
Regularization	11^{th}	1	4	N/A	N/A	ratio	100	222.78 sec

Table 5.11: Time measurements for range and scene flow for the 11^{th} image of Sequence 1.

Chapter 6

Conclusions and Future Work

We draw conclusions based on our work. Future enhancements to the work reported in this thesis is also outlined.

6.1 Conclusions

Range Flow and scene flow are the 3D optical flow on visible environmental surfaces. They are the same 3D concept, but range flow is computed from a depth map and its spatio-temporal derivatives while scene flow is computed from a disparity map (and its gradient map) as well as the 2D optical flow of the left and right images in a stereo image sequence.

We presented a scene flow algorithm and a range flow algorithm (least squares and regularization) and performed a quantitative and qualitative analysis on them. We tested the 2 algorithms on synthetic and real car driving sequences.

For range flow, we saw that least squares performed better than regularization. However, these algorithms both need to be put inside a robust framework to deal with outliers. We also found that there were high magnitude error. This is probably due to poor differentiation.

Because we have incomplete results we are not able to say definitely which algorithm is best.

6.2 Future Work

Future work includes improving the spatio-temporal derivatives so that the magnitude error is not that high. We hope to find better stereo algorithms to check our program, such as the ones on the Middlebury website. We also will put the least squares and the regularization range flow algorithms in a robust framework to deal with outliers. We also hope to get the Wedel method to work at all levels in the pyramid.

Bibliography

[1] Baris Baykant Alagoz. Obtaining depth maps from color images by region based stereo matching algorithms. *OncuBilim Algorithm And Systems Labs*, 08, 2008.

[2] J. Barron. Experience with 3d optical flow on gated mri cardiac datasets. In *Proceedings of the 1st Canadian Conference on Computer and Robot Vision*, pages 370–377, 2004.

[3] J. Barron, D. Fleet, and S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[4] J.L. Barron and H. Spies. Quantitative regularized range flow. In *Vision Interface, VI2000*, pages 203–210, 2000.

[5] James R. Bergen, P. Anandan, Th J. Hanna, and Rajesh Hingorani. Hierarchical model-based motion estimation. In *ECCV*, pages 237–252. Springer-Verlag, 1992.

[6] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *Proceedings of the 8th European Conference on Computer Vision*, pages 25–36, 2004.

[7] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems withapplications to imaging. *Journal of Mathematical Imaging and Vision*, 40:120–145, 2011.

[8] M. Faisal and J. Barron. High accuracy optical flow method based on a theory for warping: Implementation and qualitative/quantitative evaluation. In *Proceedings of the* 4^{th} *International Conference on Image Analysis and Recognition*, pages 513–525, 2007.

[9] R. Gonzalez, R. Woods, and S. Eddins. *Digital Image Processing Using MATLAB*. Prentice-Hall, Inc., 2003.

[10] B. Jahne Hagen Spies and J.L. Barron. Dense range flow from depth and intensity data. In *International Conference on Pattern Recognition (ICPR2000)*, volume 1, pages 131–134, September 2000.

[11] D. Hanselman and B. Littlefield. *Mastering MATLAB 7*. Pearson Prentice Hall, 2005.

[12] M. Harville, A. Rahimi, T.J. Darrell, G.G. Gordon, and J.I. Woodfill. 3d pose tracking with linear depth and brightness constraints. In *ICCV99*, pages 206–213, 1999.

[13] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, 1981.

[14] Frédéric Huguet and Frédéric Devernay. A variational method for scene flow estimation from stereo sequences. In *Eleventh International Conference on Computer Vision*, Rio de Janeiro, Brasil, October 2007. IEEE.

[15] M. Isard and J. MacCormick. Dense motion and disparity estimation via loopy belief propagation. In *ACCV (2)*, volume 3852 of *Lecture Notes in Computer Science*, pages 32–41. Springer, 2006.

[16] Zhifeng Liu and Reinhard Klette. Performance evaluation of stereo and motion analysis on rectified image sequences. Technical Report CITR-TR-207, Com-

puter Science Department, CITR, The University of Auckland, Auckland, New Zealand, 2007.

[17] H.C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Proceedings of the Royal Society of London B, Biology Sciences*, 208(1173):385–397, July 1980.

[18] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[19] I. Patras, E.A. Hendriks, and G. Tziritas. A joint motion/disparity estimation method for the construction of stereo interpolated images in stereoscopic image sequences. In *H.E. Bal, H. Corporaal, P.P. Jonker, J.F.M. Tonino, (eds.); Proceedings of the third annual conference of the Advanced School for Computing and Imaging*, Heijen, The Netherlands, 1997.

[20] C. Rabe, U. Franke, and S. Gehrig. Fast detection of moving objects in complex scenarios. In *IEEE Intelligent Vehicles Symposium*, pages 398–409, June 2007.

[21] E. P. Simoncelli. Design of Multi-Dimensional Derivative Filters. In *ICIP (1)*, pages 790–794, 1994.

[22] H. Spies, Bernd Jahne, and John L. Barron. Regularized range flow. In *European Conference on Computer Vision, ECCV2000*, volume 1842 and 1843 of *Lecture Notes in Computer Science (LNCS), Ed. David Vernon*, pages 785–799. Springer, 2000.

[23] Hagen Spies, Horst Hauecker, Bernd Jahne, and John L. Barron. Differential range flow estimation. In *Mustererkennung 1999, 21. DAGM-Symposium*, pages 309–316, London, UK, 1999. Springer-Verlag.

[24] Hagen Spies, Bernd Jahne, and John L. Barron. Range flow estimation. *Computer Vision and Image Understanding (CVIU2002)*, 85(3):209–231, March 2002.

[25] Tobi Vaudrey, Clemens Rabe, Reinhard Klette, and James Milburn. Differences between stereo and motion behaviour on synthetic and real-world stereo sequences. In *23rd International Conference of Image and Vision Computing New Zealand (IVCNZ '08)*, pages 1–6, 2008.

[26] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Proceedings of the 7th International Conference on Computer Vision*, volume 2, pages 722 – 729, September 1999.

[27] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):475–480, March 2005.

[28] Andreas Wedel, Thomas Brox, Tobi Vaudrey, Clemens Rabe, Uwe Franke, and Daniel Cremers. Stereoscopic scene flow computation for 3d motion understanding. *International Journal of Computer Vision (IJCV)*, October 2010.

[29] Andreas Wedel, Clemens Rabe, Tobi Vaudrey, Thomas Brox, Uwe Franke, and Daniel Cremers. Efficient dense scene flow from sparse or dense stereo data. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision*, pages 739–751, 2008.

[30] Ye Zhang, Chandra Kambhamettu, and Ra Kambhamettu. On 3d scene flow and structure estimation. In *In IEEE Conf. on Computer Vision and Pattern Recognition*, pages 778–785, 2001.